BORN

BlackBox

Server	BORN
Author	Johnson, Chris
Created	02 November 2016 08:16
File Path	C:\Users\cjohnson\Documents\My Database Documentation\BlackBox-2016-11-02T08-16-05.pdf

Table of Contents

Table of Contents 2
BORN 8
User databases 9
BlackBox Database 10
Tables 11
[dbo].[ExecForEachDBLogs] 14
[dbo].[GLMapping20151020] 15
[History].[ArchiveApSupplier27OCT2016112011060] 16
[History].[ArchiveArCustomer27OCT2016112011030] 20
[History].[ArchiveAssetMaster27OCT2016112011047] 45
[History].[ArchiveBomStructure27OCT2016112011093] 49
[History].[ArchiveExecForEachDBErrorLogs27OCT2016112011107] 53
[History].[ArchiveExecForEachDBLogs27OCT2016112011103] 54
[History].[ArchiveInvMaster27OCT2016112010997] 55
[History].[ArchiveInvWarehouse27OCT2016112011133] 65
[History].[ArchivePorMasterDetail27OCT2016112011023] 67
[History].[ArchivePorMasterHdr27OCT2016112011037] 72
[History].[ArchiveRedTagLogs27OCT2016112011100] 74
[History].[ArchiveReqDetail27OCT2016112011070] 76
[History].[ArchiveReqHeader27OCT2016112011077] 79
[History].[ArchiveSorMaster27OCT2016112011007] 81
[History].[ArchiveTblCurrency27OCT2016112011017] 84
[History].[ArchiveWipInspect27OCT2016112011087] 86
[History].[ArchiveWipJobAllLab27OCT2016112011113] 88
[History].[ArchiveWipJobAllMat27OCT2016112011123] 90
[History].[ArchiveWipMaster27OCT2016112011140] 91
[History].[RedTagLogs] 94
[Lookups].[AdmTransactionIDs] 96
[Lookups].[ApJnlDistribExpenseType] 97
[Lookups].[ApSupplier] 98
[Lookups].[ArInvoicePayTrnType] 99
[Lookups].[BankBalances] 100
[Lookups].[Bin] 102
[Lookups].[BudgetType] 103
[Lookups].[Buyers] 104
[Lookups].[CompanyNames] 105
[Lookups].[CurrencyRates] 107
[Lookups].[GenJournalCtlJnlSource] 109
[Lookups].[GenJournalCtlSource] 110
[Lookups].[GenJournalDetailSource] 111

[Lookups].[GenJournalType] 112
[Lookups].[GenTransactionSource] 113
[Lookups].[GlAccountCode] 114
[Lookups].[GLAccountType] 115
[Lookups].[GlAnalysisCategory] 116
[Lookups].[GlExpenseCode] 117
[Lookups].[GLMapping] 118
[Lookups].[HolidayDays] 120
[Lookups].[InvMaster_PartCategory] 121
[Lookups].[JnlPostingType] 122
[Lookups].[JnlStatus] 123
[Lookups].[LedgerGroupMaps] 124
[Lookups].[LotTransactionTrnType] 125
[Lookups].[MCompleteFlag] 126
[Lookups].[PorLineType] 127
[Lookups].[ProductClass] 128
[Lookups].[ProformaDocTypes] 129
[Lookups].[PurchaseOrderInvoiceMapping] 130
[Lookups].[PurchaseOrderStatus] 131
[Lookups].[PurchaseOrderTaxStatus] 132
[Lookups].[PurchaseOrderType] 133
[Lookups].[RedTagsUsedByType] 134
[Lookups].[ReportIndexUserMaps] 135
[Lookups].[ReqBuyers] 136
[Lookups].[ReqnStatus] 137
[Lookups].[SalesOrderDocumentType] 138
[Lookups].[SalesOrderLineType] 139
[Lookups].[SalesOrderStatus] 140
[Lookups].[StockCode] 141
[Lookups].[TrnTypeAmountModifier] 142
[Lookups].[Warehouse] 143
[Process].[GrnInvoiceDetails] 144
[Process].[RefreshTableTimes] 145
[Process].[Status_WipSubJobStock] 146
[Process].[SysproTransactionsLogged] 147
[Report].[GrnInvoiceDetails] 151
[Report].[Results_WipSubJobStock] 153
Views 156
[dbo].[vw_APSuppliersWithActivePO] 157
[Lookups].[StockCodes_Bought] 158
[Lookups].[vw_AnalysisCategoryDistinct] 159

[Lookups].[vw_APSuppliersWithActivePO] 160
[Lookups].[vw_DistinctBuyerList] 161
[Lookups].[vw_StockCodes_Bought_ActivePO] 162
Stored Procedures 163
[Process].[ExecForEachDB] 169
[Process].[ExecForEachDB_WithTableCheck] 174
[Process].[UspAlter_AddColumn] 179
[Process].[UspAlter_CheckAndAddColumns] 180
[Process].[UspAlter_TableName] 183
[Process].[UspCreate_Table] 186
[Process].[UspCreateAmend_HistoryTables] 187
[Process].[UspInsert_RedTagLogs] 194
[Process].[UspLoad_LoadController] 200
[Process].[UspPopulate_HistoryTables1] 202
[Process].[UspPopulate_UnpivotHistory] 217
[Process].[UspUpdate_ApSupplier] 223
[Process].[UspUpdate_ArInvoicePayTrnType] 227
[Process].[UspUpdate_BankBalances] 230
[Process].[UspUpdate_Bin] 234
[Process].[UspUpdate_BudgetType] 238
[Process].[UspUpdate_Buyers] 241
[Process].[UspUpdate_CompanyNames] 245
[Process].[UspUpdate_CurrencyRates] 252
[Process].[UspUpdate_GenJournalCtlJnlSource] 261
[Process].[UspUpdate_GenTransactionSource] 264
[Process].[UspUpdate_GlAccountCode] 268
[Process].[UspUpdate_GLAccountType] 272
[Process].[UspUpdate_GlAnalysisCategories] 276
[Process].[UspUpdate_GlExpenseCode] 280
[Process].[UspUpdate_InvMaster_PartCategory] 284
[Process].[UspUpdate_MCompleteFlag] 288
[Process].[UspUpdate_PorLineType] 292
[Process].[UspUpdate_ProductClass] 295
[Process].[UspUpdate_PurchaseOrderInvoiceMapping] 299
[Process].[UspUpdate_PurchaseOrderStatus] 303
[Process].[UspUpdate_PurchaseOrderTaxStatus] 307
[Process].[UspUpdate_PurchaseOrderType] 311
[Process].[UspUpdate_ReqBuyers] 315
[Process].[UspUpdate_ReqnStatus] 319
[Process].[UspUpdate_SalesOrderDocumentType] 323
[Process].[UspUpdate_SalesOrderLineType] 326

[Process].[UspUpdate_SalesOrderStatus] 330
[Process].[UspUpdate_StockCode] 335
[Process].[UspUpdate_TrnTypeModifier] 339
[Process].[UspUpdate_Warehouse] 343
[Report].[Results_AllCurrencyRates] 347
[Report].[Results_BudgetTest] 349
[Report].[UspResults_AccPayable_AgedInvoices] 357
[Report].[UspResults_AllBankBalances] 363
[Report].[UspResults_AllPosAndReqs] 372
[Report].[UspResults_AllSalesOrders] 387
[Report].[UspResults_AllSuppliers] 397
[Report].[UspResults_AmendmentJnl] 400
[Report].[UspResults_ApAgingInvoices] 404
[Report].[UspResults_APDaysToPayment] 416
[Report].[UspResults_ApGlDisburse] 422
[Report].[UspResults_ApInvoice] 426
[Report].[UspResults_ApInvoicesWithPaymentDetails] 430
[Report].[UspResults_ApJnlGLDistrs] 437
[Report].[UspResults_ApSupplierNar] 441
[Report].[UspResults_ApSuppNarr] 445
[Report].[UspResults_ArCustomers] 447
[Report].[UspResults_ARDaysToPayment] 451
[Report].[UspResults_AvailableLots] 458
[Report].[UspResults_AvailableLots_AmountRequired] 465
[Report].[UspResults_AvailableStock] 471
[Report].[UspResults_BudgetsActuals] 478
[Report].[UspResults_BudgetsVsActuals] 482
[Report].[UspResults_ClosingBalancesInterco] 488
[Report].[UspResults_CompaniesCurrencyLatestExchangeRates] 492
[Report].[UspResults_CompanyTransactions] 494
[Report].[UspResults_ExecForEachDBLogs] 502
[Report].[UspResults_FixedAssets] 503
[Report].[UspResults_GeneralLedgerControlStats] 510
[Report].[UspResults_GenJournalDetails] 512
[Report].[UspResults_GenJournalEntries] 520
[Report].[UspResults_GenledgerJournalsGrouped] 533
[Report].[UspResults_GenMasterMapping] 539
[Report].[UspResults_GL_Actuals] 543
[Report].[UspResults_GL_Budgets] 546
[Report].[UspResults_GL_Codes] 547
[Report].[UspResults_GLMovements] 549

[Report].[UspResults_GLMovementsIntercoPL] 555
[Report].[UspResults_GLMovementsPBLRevCos] 558
[Report].[UspResults_GrnInvoiceDetails] 562
[Report].[UspResults_InventoryInInspection] 573
[Report].[UspResults_InventoryInspectionTimes] 579
[Report].[UspResults_InvoiceRunVerify] 589
[Report].[UspResults_JobHeader] 602
[Report].[UspResults_JobHeader_AllJobs] 614
[Report].[UspResults_JobStockDetails] 628
[Report].[UspResults_JournalDetails] 633
[Report].[UspResults_Labour] 639
[Report].[UspResults_LabourDetails] 643
[Report].[UspResults_LabourGlCodes] 648
[Report].[UspResults_LabourLoggedPerISOWeek] 651
[Report].[UspResults_ListAllCompanies] 655
[Report].[UspResults_LotsRetesting] 658
[Report].[UspResults_LotTraceability] 667
[Report].[UspResults_LotTraceability_WithRuntimes] 676
[Report].[UspResults_MovementsTrialBalances] 686
[Report].[UspResults_OpenPurchaseOrderStock] 690
[Report].[UspResults_OutstandingPurchaseOrders] 696
[Report].[UspResults_PaymentRunVerify] 703
[Report].[UspResults_PosNoReqs] 715
[Report].[UspResults_PurchaseOrderChanges] 720
[Report].[UspResults_PurchaseOrderDetails] 726
[Report].[UspResults_PurchaseOrders] 736
[Report].[UspResults_PurchaseOrdersHistory] 745
[Report].[UspResults_PurchaseOrdersInvoices] 751
[Report].[UspResults_PurchaseOrdersOpen] 761
[Report].[UspResults_PurchaseOrderWithHistory] 770
[Report].[UspResults_ReceivedLotJob] 777
[Report].[UspResults_RedTagDetails] 780
[Report].[UspResults_ReportSysRefresh_RedTagLogs] 782
[Report].[UspResults_ReqsAfterInvoices] 788
[Report].[UspResults_RequisitionStatus] 798
[Report].[UspResults_RequisitionUsers] 804
[Report].[UspResults_ReservedLots] 811
[Report].[UspResults_SalesOrderKPI] 817
[Report].[UspResults_SalesOrders] 828
[Report].[UspResults_SalesOrderStats] 836
[Report].[UspResults_ScrapCosts] 844

[Report].[UspResults_SearchForMissingIndexes] 850
[Report].[UspResults_StockDispatches] 853
[Report].[UspResults_StockLevels] 863
[Report].[UspResults_StockMovements] 870
[Report].[UspResults_StockOutput] 878
[Report].[UspResults_StockOutputDispatches] 888
[Report].[UspResults_StockOutputDispatches2] 892
[Report].[UspResults_StockWithRunTimes] 910
[Report].[UspResults_Template] 917
[Report].[UspResults_TrialBalance] 920
[Report].[UspResults_UnpaidAssets] 924
[Report].[UspResults_UnpaidGRNAssets] 937
[Report].[UspResults_WipStockRequiredPerJob] 946
[Report].[UspResults_WipSubJobStock] 960
[Report].[UspResultsRefresh_TableTimes] 979
[Reports].[UspResults_SearchForProcedures] 985
[Reports].[UspResults_SearchForText] 988
[Review].[UspResults_CompanyDuplicates] 995
[Review].[UspResults_CompanyNamesMissing] 997
[Review].[UspResults_CurrencyRatesMissing] 999
[Review].[UspResults_DbDiffs] 1001
[Review].[UspResults_LastUpdatedTimes] 1005
[Review].[UspResults_LogStats] 1007
[Review].[UspResults_LookupDuplicates] 1009
[Review].[UspResults_ProcsMostRun] 1018
[Review].[UspResults_ReportsDevelopmentRun] 1021
[Review].[UspResults_ReportsMostRun] 1024
[Review].[UspResults_SysproTransactionsCheck] 1027
[Review].[UspResults_SysproTransactionsTrending] 1028
[Review].[UspResults_WarehousesIssueFrom] 1030
Table-valued Functions 1033
[dbo].[udf_SplitString] 1034
[dbo].[UdfResults_NumberRange] 1036
Scalar-valued Functions 1039
[Process].[Udf_WorkingDays] 1040

BORN

Databases(1)

- BlackBox

User databases

Databases(1)

- BlackBox

BlackBox Database

Database Properties

Property	Value
SQL Server Version	SQL Server 2014
Compatibility Level	SQL Server 2008
Database Encryption Enabled	False
Last backup time	-
Last log backup time	-
Creation date	Dec 11 2015
Users	5
Database size	2537.19 MB
Unallocated space	431.80 MB

Tables

Objects

Name
dbo.ExecForEachDBLogs place to capture details of exec for each db
dbo.GLMapping20151020 General Ledger mapping as provided by finance
History.ArchiveApSupplier27OCT2016112011060
History.ArchiveArCustomer27OCT2016112011030 Logs from ArCustomer change logs
History.ArchiveAssetMaster27OCT2016112011047 Logs from AssetManager change logs
History.ArchiveBomStructure27OCT2016112011093 Logs from BOMStructure change logs
History.ArchiveExecForEachDBErrorLogs27OCT2016112011107 place to capture details of errors generated by sp ExecForEachDB
History.ArchiveExecForEachDBLogs27OCT2016112011103 place to capture details of runs generated by sp ExecForEachDB
History.ArchiveInvMaster27OCT2016112010997 Logs from InvMaster change logs
History.ArchiveInvWarehouse27OCT2016112011133
History.ArchivePorMasterDetail27OCT2016112011023
History.ArchivePorMasterHdr27OCT2016112011037
History.ArchiveRedTagLogs27OCT2016112011100 history of reports run
History.ArchiveReqDetail27OCT2016112011070
History.ArchiveReqHeader27OCT2016112011077
History.ArchiveSorMaster27OCT2016112011007
History.ArchiveTblCurrency27OCT2016112011017 Logs from TblCurrency change logs
History.ArchiveWipInspect27OCT2016112011087
History.ArchiveWipJobAllLab27OCT2016112011113
History.ArchiveWipJobAllMat27OCT2016112011123
History.ArchiveWipMaster27OCT2016112011140
History.RedTagLogs
Lookups.AdmTransactionIDs AdmTransaction Descriptions for use in reports
Lookups.ApJnlDistribExpenseType

Author: Johnson, Chris

ApJnlDistribExpenseType Descriptions for use in reports
Lookups.ApSupplier List of all suppliers
Lookups.ArInvoicePayTrnType ArInvoicePay TrnType Descriptions for use in reports
Lookups.BankBalances History of BankBalances
Lookups.Bin list of bins to be used in reports
Lookups.BudgetType list of budget types to be used in reports
Lookups.Buyers list of all buyers
Lookups.CompanyNames List of company names used in reports
Lookups.CurrencyRates history of currency rates
Lookups.GenJournalCtlJnlSource Description of GenJournal CtlJnlSource
Lookups.GenJournalCtlSource
Lookups.GenJournalDetailSource Description of GenJournal DetailSource
Lookups.GenJournalType Description of GenJournal Type
Lookups.GenTransactionSource Description of Gen Transaction Source
Lookups.GlAccountCode Description of GL AccountCode
Lookups.GLAccountType Description of GL AccountType
Lookups.GlAnalysisCategory
Lookups.GlExpenseCode Gl Expense code descriptions
Lookups.GLMapping GL Mapping as provided by finance 2015
Lookups.HolidayDays list of non-working days used to calculate holidays
Lookups.InvMaster_PartCategory InvMaster Part Category description
Lookups.JnlPostingType JnlPosting Type description
Lookups.JnlStatus JnlStatus Description
Lookups.LedgerGroupMaps
Lookups.LotTransactionTrnType LotTransaction TrnType Description
Lookups.MCompleteFlag
Lookups.PorLineType
Lookups.ProductClass

Author: Johnson, Chris

Lookups.ProformaDocTypes
Lookups.PurchaseOrderInvoiceMapping
Lookups.PurchaseOrderStatus
Lookups.PurchaseOrderTaxStatus
Lookups.PurchaseOrderType
Lookups.RedTagsUsedByType
Lookups.ReportIndexUserMaps
Lookups.ReqBuyers
Lookups.ReqnStatus
Lookups.SalesOrderDocumentType
Lookups.SalesOrderLineType
Lookups.SalesOrderStatus
Lookups.StockCode
Lookups.TrnTypeAmountModifier
Lookups.Warehouse
Process.GrnInvoiceDetails
Process.RefreshTableTimes
Process.Status_WipSubJobStock
Process.SysproTransactionsLogged
Report.GrnInvoiceDetails
Report.Results_WipSubJobStock

[dbo].[ExecForEachDBLogs]

MS_Description

place to capture details of exec for each db

Columns

Name	Data Type	Max Length (Bytes)	Allow Nulls	Identity	Default
LogID	bigint	8	False	1 - 1	
LogTime	datetime2	8	True		(getdate())
Cmd	nvarchar(2000)	4000	True		

SQL Script

```
CREATE TABLE [dbo].[ExecForEachDBLogs]
(
[LogID] [bigint] NOT NULL IDENTITY(1, 1),
[LogTime] [datetime2] NULL CONSTRAINT [DF__ExecForEa__LogTi__1269A02C] DEFAULT
(getdate()),
[Cmd] [nvarchar] (2000) COLLATE Latin1_General_BIN NULL
) ON [PRIMARY]
GO
EXEC sp_addextendedproperty N'MS_Description', N'place to capture details of exec
for each db', 'SCHEMA', N'dbo', 'TABLE', N'ExecForEachDBLogs', NULL, NULL
GO
```

Author: Johnson, Chris

[dbo].[GLMapping20151020]

MS_Description

General Ledger mapping as provided by finance

Columns

Name	Data Type	Max Length (Bytes)	Allow Nulls
GlCode	varchar(150)	150	True
GlStart	char(3)	3	True
GlMid	char(5)	5	True
GlEnd	char(3)	3	True
Mid1	char(3)	3	True
Mid2	char(2)	2	True
Company	varchar(10)	10	True
GlDescription	varchar(150)	150	True
Mapping1	varchar(255)	255	True
Mapping2	varchar(255)	255	True
Mapping3	varchar(255)	255	True
Mapping4	varchar(255)	255	True
Mapping5	varchar(255)	255	True

SQL Script

```
CREATE TABLE [dbo].[GLMapping20151020]
(
[GlCode] [varchar] (150) COLLATE Latin1_General_BIN NULL,
[GlStart] [char] (3) COLLATE Latin1_General_BIN NULL,
[GlMid] [char] (5) COLLATE Latin1_General_BIN NULL,
[GlEnd] [char] (3) COLLATE Latin1_General_BIN NULL,
[Mid1] [char] (3) COLLATE Latin1_General_BIN NULL,
[Mid2] [char] (2) COLLATE Latin1_General_BIN NULL,
[Company] [varchar] (10) COLLATE Latin1_General_BIN NULL,
[GlDescription] [varchar] (150) COLLATE Latin1_General_BIN NULL,
[Mapping1] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[Mapping2] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[Mapping3] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[Mapping4] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[Mapping5] [varchar] (255) COLLATE Latin1_General_BIN NULL
) ON [PRIMARY]
GO
EXEC sp_addextendedproperty N'MS_Description', N'General Ledger mapping as provided
by finance', 'SCHEMA', N'dbo', 'TABLE', N'GLMapping20151020', NULL, NULL
GO
```

Author: Johnson, Chris

[History].[ArchiveApSupplier27OCT2016112011060]

Columns

Name	Data Type	Max Length (Bytes)	Allow Nulls	Identity
AID	int	4	False	1 - 1
TransactionDescription	varchar(150)	150	True	
SignatureDatetime	datetime2	8	False	
Operator	varchar(20)	20	False	
ProgramName	varchar(20)	20	False	
Ranking	bigint	8	True	
ItemKey	varchar(150)	150	False	
DatabaseName	varchar(150)	150	False	
BANKBEFORE	varchar(255)	255	True	
BANKBRANCHCODE	varchar(255)	255	True	
BRANCHBEFORE	varchar(255)	255	True	
CURRENTCOMPANYID	varchar(255)	255	True	
CURRENTOPERATINGSYSTEMDATE	datetime	8	True	
CURRENTOPERATINGSYSTEMTIME	varchar(255)	255	True	
OPERATORCODE	varchar(255)	255	True	
OPERATOREMAILADDRESS	varchar(255)	255	True	
OPERATORNAME	varchar(255)	255	True	
SUPPLIERBANK	varchar(255)	255	True	
SUPPLIERBANKACCOUNT	varchar(255)	255	True	
SUPPLIERBRANCH	varchar(255)	255	True	
SUPPLIERCODE	varchar(255)	255	True	
SUPPLIERNAME	varchar(255)	255	True	
SUPPLIERSHORTNAME	varchar(255)	255	True	
APINVOICEDISCOUNT	varchar(255)	255	True	
APINVOICETERMSCODE	varchar(255)	255	True	
BANKACCOUNTBEFORE	varchar(255)	255	True	
BANKBRANCHBEFORE	varchar(255)	255	True	
CONDITIONDESCRIPTION	varchar(255)	255	True	
CURRENCY	varchar(255)	255	True	
CURRENTCOMPANYDATE	datetime	8	True	
CURRENTCOMPANYNAME	varchar(255)	255	True	
DEFAULTTAXCODE	varchar(255)	255	True	
DELIVERYTERMSEU	varchar(255)	255	True	
EFTBANKACCOUNTTYPE	varchar(255)	255	True	
GEOGRAPHICAREA	varchar(255)	255	True	
GRNMATCHINGALLOWEDBEFORE	varchar(255)	255	True	
GRNMATCHINGREQUIRED	varchar(255)	255	True	

Author: Johnson, Chris

LANGUAGECODE	varchar(255)	255	True	
LCTREQUIRED	varchar(255)	255	True	
LCTREQUIREDBEFORE	varchar(255)	255	True	
LOCALCURRENCY	varchar(255)	255	True	
MINPURCHASEORDERMASS	float	8	True	
MINPURCHASEORDERVALUELOCAL	float	8	True	
MINPURCHASEORDERVOLUME	float	8	True	
NATIONALITYCODE	varchar(255)	255	True	
ONHOLDFLAGBEFORE	varchar(255)	255	True	
OPERATORCURRENTGROUPCODE	varchar(255)	255	True	
OPERATORLOCATION	varchar(255)	255	True	
OPERATORPRIMARYGROUPCODE	varchar(255)	255	True	
PAYBYEFT	varchar(255)	255	True	
PAYBYEFTBEFORE	varchar(255)	255	True	
PAYE1099REFERENCE	varchar(255)	255	True	
POALLOWEDBEFORE	varchar(255)	255	True	
PURCHASEORDERLINEDISCOUNT	varchar(255)	255	True	
PURCHASEORDERSALLOWED	varchar(255)	255	True	
SOURCEAPPLICATION	varchar(255)	255	True	
SUPPLIERCHECKNAME	varchar(255)	255	True	
SUPPLIERCHECKNAMEBEFORE	varchar(255)	255	True	
SUPPLIERCLASS	varchar(255)	255	True	
SUPPLIERONHOLDFLAG	varchar(255)	255	True	
SUPPLIERTYPE	varchar(255)	255	True	
TAXREGISTRATIONNUMBEREU	varchar(255)	255	True	
TERMSCODEBEFORE	varchar(255)	255	True	
USERDEFINEDFIELD1	varchar(255)	255	True	
USERDEFINEDFIELD2	varchar(255)	255	True	
WITHHOLDINGTAXCODE	varchar(255)	255	True	
WITHHOLDINGTAXID	varchar(255)	255	True	

SQL Script

```
CREATE TABLE [History].[ArchiveApSupplier27OCT2016112011060]
(
[AID] [int] NOT NULL IDENTITY(1, 1),
[TransactionDescription] [varchar] (150) COLLATE Latin1_General_BIN NULL,
[SignatureDatetime] [datetime2] NOT NULL,
[Operator] [varchar] (20) COLLATE Latin1_General_BIN NOT NULL,
[ProgramName] [varchar] (20) COLLATE Latin1_General_BIN NOT NULL,
[Ranking] [bigint] NULL,
[ItemKey] [varchar] (150) COLLATE Latin1_General_BIN NOT NULL,
[DatabaseName] [varchar] (150) COLLATE Latin1_General_BIN NOT NULL,
[BANKBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[BANKBRANCHCODE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[BRANCHBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[CURRENTCOMPANYID] [varchar] (255) COLLATE Latin1_General_BIN NULL,
```

Author: Johnson, Chris

```
[CURRENTOPERATINGSYSTEMDATE] [datetime] NULL,
[CURRENTOPERATINGSYSTEMTIME] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[OPERATORCODE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[OPERATOREMAILADDRESS] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[OPERATORNAME] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[SUPPLIERBANK] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[SUPPLIERBANKACCOUNT] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[SUPPLIERBRANCH] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[SUPPLIERCODE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[SUPPLIERNAME] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[SUPPLIERSHORTNAME] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[APINVOICEDISCOUNT] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[APINVOICETERMSCODE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[BANKACCOUNTBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[BANKBRANCHBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[CONDITIONDESCRIPTION] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[CURRENCY] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[CURRENTCOMPANYDATE] [datetime] NULL,
[CURRENTCOMPANYNAME] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[DEFAULTTAXCODE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[DELIVERYTERMSEU] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[EFTBANKACCOUNTTYPE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[GEOGRAPHICAREA] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[GRNMATCHINGALLOWEDBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[GRNMATCHINGREQUIRED] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[LANGUAGECODE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[LCTREQUIRED] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[LCTREQUIREDBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[LOCALCURRENCY] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[MINPURCHASEORDERMASS] [float] NULL,
[MINPURCHASEORDERVALUELOCAL] [float] NULL,
[MINPURCHASEORDERVOLUME] [float] NULL,
[NATIONALITYCODE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[ONHOLDFLAGBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[OPERATORCURRENTGROUPCODE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[OPERATORLOCATION] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[OPERATORPRIMARYGROUPCODE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[PAYBYEFT] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[PAYBYEFTBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[PAYE1099REFERENCE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[POALLOWEDBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[PURCHASEORDERLINEDISCOUNT] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[PURCHASEORDERSALLOWED] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[SOURCEAPPLICATION] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[SUPPLIERCHECKNAME] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[SUPPLIERCHECKNAMEBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[SUPPLIERCLASS] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[SUPPLIERONHOLDFLAG] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[SUPPLIERTYPE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[TAXREGISTRATIONNUMBEREU] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[TERMSCODEBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[USERDEFINEDFIELD1] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[USERDEFINEDFIELD2] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[WITHHOLDINGTAXCODE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[WITHHOLDINGTAXID] [varchar] (255) COLLATE Latin1_General_BIN NULL
) ON [PRIMARY]
GO
```

Author: Johnson, Chris

Uses

[History]

Author: Johnson, Chris

[History].[ArchiveArCustomer27OCT2016112011030]

MS_Description

Logs from ArCustomer change logs

Columns

Name	Data Type	Max Length (Bytes)	Allow Nulls
TransactionDescription	varchar(150)	150	True
DatabaseName	varchar(150)	150	False
SignatureDateTime	datetime2	8	False
Operator	varchar(20)	20	False
ItemKey	varchar(150)	150	False
ComputerName	varchar(150)	150	True
ProgramName	varchar(100)	100	False
ConditionName	varchar(15)	15	True
AlreadyEntered	bit	1	True
A2STECHNOLOGIESAFTER	varchar(255)	255	True
A2STECHNOLOGIESBEFORE	varchar(255)	255	True
AALTOSCIENTIFICLIMITEDAFTER	varchar(255)	255	True
AALTOSCIENTIFICLIMITEDBEFORE	varchar(255)	255	True
ABBOMAXINC.AFTER	varchar(255)	255	True
ABBOMAXINC.BEFORE	varchar(255)	255	True
ABBOTTLABORATORIESAFTER	varchar(255)	255	True
ABBOTTLABORATORIESBEFORE	varchar(255)	255	True
ABBOTTPOINTOFCARECANADALTDAFTER	varchar(255)	255	True
ABBOTTPOINTOFCARECANADALTDBEFORE	varchar(255)	255	True
ABBVIEBIORESEARCHCENTERINCAFTER	varchar(255)	255	True
ABBVIEBIORESEARCHCENTERINCBEFORE	varchar(255)	255	True
ABSCILLCAFTER	varchar(255)	255	True
ABSCILLCBEFORE	varchar(255)	255	True
ABSOLUTEANTIBODYAFTER	varchar(255)	255	True
ABSOLUTEANTIBODYBEFORE	varchar(255)	255	True
ADBIOTECHSOLUTIONSAFTER	varchar(255)	255	True
ADBIOTECHSOLUTIONSBEFORE	varchar(255)	255	True
ADCBIOTECHNOLOGYAFTER	varchar(255)	255	True
ADCBIOTECHNOLOGYBEFORE	varchar(255)	255	True
ADNEXUS-ABMSR&DCOMPANY(0590)AFTER	varchar(255)	255	True
ADNEXUS-ABMSR&DCOMPANY(0590)BEFORE	varchar(255)	255	True
ADVYLTDAFTER	varchar(255)	255	True
ADVYLTDBEFORE	varchar(255)	255	True
AGGAMINBIOLOGICSAFTER	varchar(255)	255	True

Author: Johnson, Chris

AGGAMINBIOLOGICSBEFORE	varchar(255)	255	True
AJUSTHOLDINGA/SAFTER	varchar(255)	255	True
AJUSTHOLDINGA/SBEFORE	varchar(255)	255	True
ALLERGANINC.AFTER	varchar(255)	255	True
ALLERGANINC.BEFORE	varchar(255)	255	True
ALLERGANSALESLLCAFTER	varchar(255)	255	True
ALLERGANSALESLLCBEFORE	varchar(255)	255	True
ALTMETHODFLAGAFTER	varchar(255)	255	True
ALTMETHODFLAGBEFORE	varchar(255)	255	True
AMBRX,INC.AFTER	varchar(255)	255	True
AMBRX,INC.BEFORE	varchar(255)	255	True
AMGENAFTER	varchar(255)	255	True
AMGENBEFORE	varchar(255)	255	True
AMGENRESEARCH(MUNICH)GMBHAFTER	varchar(255)	255	True
AMGENRESEARCH(MUNICH)GMBHBEFORE	varchar(255)	255	True
ANASPECINCAFTER	varchar(255)	255	True
ANASPECINCBEFORE	varchar(255)	255	True
AREAAFTER	varchar(255)	255	True
AREABEFORE	varchar(255)	255	True
ARKRAYAFTER	varchar(255)	255	True
ARKRAYBEFORE	varchar(255)	255	True
ASPENPHARMACAREAFTER	varchar(255)	255	True
ASPENPHARMACAREBEFORE	varchar(255)	255	True
ATOLLGMBHAFTER	varchar(255)	255	True
ATOLLGMBHBEFORE	varchar(255)	255	True
AUXILIUMPHARMACEUTICALSINCAFTER	varchar(255)	255	True
AUXILIUMPHARMACEUTICALSINCBEFORE	varchar(255)	255	True
AVAXIABIOLOGICSINCAFTER	varchar(255)	255	True
AVAXIABIOLOGICSINCBEFORE	varchar(255)	255	True
AVIDBIOSERVICESAFTER	varchar(255)	255	True
AVIDBIOSERVICESBEFORE	varchar(255)	255	True
AVITIDE,INC.AFTER	varchar(255)	255	True
AVITIDE,INC.BEFORE	varchar(255)	255	True
B-BRIDGEINTERNATIONALINCAFTER	varchar(255)	255	True
B-BRIDGEINTERNATIONALINCBEFORE	varchar(255)	255	True
BACB.V.AFTER	varchar(255)	255	True
BACB.V.BEFORE	varchar(255)	255	True
BAYLORCOLLEGEOFMEDICINEAFTER	varchar(255)	255	True
BAYLORCOLLEGEOFMEDICINEBEFORE	varchar(255)	255	True
BEIJINGZHONGYUANLIMITEDAFTER	varchar(255)	255	True
BEIJINGZHONGYUANLIMITEDBEFORE	varchar(255)	255	True
BIO-RADABDSEROTECGMBHAFTER	varchar(255)	255	True
BIO-RADABDSEROTECGMBHBEFORE	varchar(255)	255	True
BIO-RADDIAGNOSTICSINCAFTER	varchar(255)	255	True

BIO-RADDIAGNOSTICSINCBEFORE	varchar(255)	255	True
BIO-RADLABORATORIESDEESIDELTDAFTER	varchar(255)	255	True
BIO-RADLABORATORIESDEESIDELTDBEFORE	varchar(255)	255	True
BIO-RADLABORATORIESINCAFTER	varchar(255)	255	True
BIO-RADLABORATORIESINCBEFORE	varchar(255)	255	True
BIO-TECHNOLOGYGENERAL(ISRAEL)LTDAFTER	varchar(255)	255	True
BIO-TECHNOLOGYGENERAL(ISRAEL)LTDBEFORE	varchar(255)	255	True
BIOANALYTICMACEIJSTOPAAFTER	varchar(255)	255	True
BIOANALYTICMACEIJSTOPABEFORE	varchar(255)	255	True
BIOCONRESEARCHLIMITEDSEZUNITAFTER	varchar(255)	255	True
BIOCONRESEARCHLIMITEDSEZUNITBEFORE	varchar(255)	255	True
BIOINVENTINTERNATIONALABAFTER	varchar(255)	255	True
BIOINVENTINTERNATIONALABBEFORE	varchar(255)	255	True
BIONISYSRDCORPAFTER	varchar(255)	255	True
BIONISYSRDCORPBEFORE	varchar(255)	255	True
BIORADAFTER	varchar(255)	255	True
BIORADBEFORE	varchar(255)	255	True
BIORADLABORATORIESAFTER	varchar(255)	255	True
BIORADLABORATORIESBEFORE	varchar(255)	255	True
BIORESOURCETECHNOLOGYAFTER	varchar(255)	255	True
BIORESOURCETECHNOLOGYBEFORE	varchar(255)	255	True
BIOTESTAGAFTER	varchar(255)	255	True
BIOTESTAGBEFORE	varchar(255)	255	True
BIOTRADERINCAFTER	varchar(255)	255	True
BIOTRADERINCBEFORE	varchar(255)	255	True
BOEHRINGERINGELHEIMAUSTRIAGMBHAFTER	varchar(255)	255	True
BOEHRINGERINGELHEIMAUSTRIAGMBHBEFORE	varchar(255)	255	True
BRANCHAFTER	varchar(255)	255	True
BRANCHBEFORE	varchar(255)	255	True
BRISTOL-MYERSSQUIBBCOAFTER	varchar(255)	255	True
BRISTOL-MYERSSQUIBBCOBEFORE	varchar(255)	255	True
BRISTOL-MYERSSQUIBBHOPKINTONAFTER	varchar(255)	255	True
BRISTOL-MYERSSQUIBBHOPKINTONBEFORE	varchar(255)	255	True
BRITISHCOLUMBIAUNIVERSITYAFTER	varchar(255)	255	True
BRITISHCOLUMBIAUNIVERSITYBEFORE	varchar(255)	255	True
CADILAPHARMACEUTICALSLTDAFTER	varchar(255)	255	True
CADILAPHARMACEUTICALSLTDBEFORE	varchar(255)	255	True
CATALENTPHARMASOLUTIONS,LLCAFTER	varchar(255)	255	True
CATALENTPHARMASOLUTIONS,LLCBEFORE	varchar(255)	255	True
CATALENTPHARMASOLUTIONSAFTER	varchar(255)	255	True
CATALENTPHARMASOLUTIONSBEFORE	varchar(255)	255	True
CEDARLANELABORATORIESUSAAFTER	varchar(255)	255	True
CEDARLANELABORATORIESUSABEFORE	varchar(255)	255	True
CELSISINTERNATIONALGMBHAFTER	varchar(255)	255	True

CELSISINTERNATIONALGMBHBEFORE	varchar(255)	255	True
CEPHALONAUSTRALIAAFTER	varchar(255)	255	True
CEPHALONAUSTRALIABEFORE	varchar(255)	255	True
CHEMDIVINCAFTER	varchar(255)	255	True
CHEMDIVINCBEFORE	varchar(255)	255	True
CHEMQUESTLIMITEDAFTER	varchar(255)	255	True
CHEMQUESTLIMITEDBEFORE	varchar(255)	255	True
CINTRADEENTERPRISE,INC.AFTER	varchar(255)	255	True
CINTRADEENTERPRISE,INC.BEFORE	varchar(255)	255	True
CJSCGENERIUMAFTER	varchar(255)	255	True
CJSCGENERIUMBEFORE	varchar(255)	255	True
COBRABIOLOGICSLTDAFTER	varchar(255)	255	True
COBRABIOLOGICSLTDBEFORE	varchar(255)	255	True
COMPANYTAXNUMBERAFTER	varchar(255)	255	True
COMPANYTAXNUMBERBEFORE	varchar(255)	255	True
CONTACTAFTER	varchar(255)	255	True
CONTACTBEFORE	varchar(255)	255	True
CONTRACTPRCREQDAFTER	varchar(255)	255	True
CONTRACTPRCREQDBEFORE	varchar(255)	255	True
COOKPHARMICAAFTER	varchar(255)	255	True
COOKPHARMICABEFORE	varchar(255)	255	True
CRESCENDOBIOLOGICSLTDAFTER	varchar(255)	255	True
CRESCENDOBIOLOGICSLTDBEFORE	varchar(255)	255	True
CRUCELLBIOLOGICSINCC/OJOHNSON&JOHNSONAFTER	varchar(255)	255	True
CRUCELLBIOLOGICSINCC/OJOHNSON&JOHNSONBEFORE	varchar(255)	255	True
CRUCELLHOLLANDBVAFTER	varchar(255)	255	True
CRUCELLHOLLANDBVBEFORE	varchar(255)	255	True
CSLBEHRINGAGAFTER	varchar(255)	255	True
CSLBEHRINGAGBEFORE	varchar(255)	255	True
CSLBEHRINGGMBHAFTER	varchar(255)	255	True
CSLBEHRINGGMBHBEFORE	varchar(255)	255	True
CSLLTDAUSTRALIAAFTER	varchar(255)	255	True
CSLLTDAUSTRALIABEFORE	varchar(255)	255	True
CURRENCYAFTER	varchar(255)	255	True
CURRENCYBEFORE	varchar(255)	255	True
CYPRESSINTERNATIONALLTDAFTER	varchar(255)	255	True
CYPRESSINTERNATIONALLTDBEFORE	varchar(255)	255	True
D.B.AITALIASRLAFTER	varchar(255)	255	True
D.B.AITALIASRLBEFORE	varchar(255)	255	True
DELIVERYTERMSAFTER	varchar(255)	255	True
DELIVERYTERMSBEFORE	varchar(255)	255	True
DISCIENCEAFTER	varchar(255)	255	True
DISCIENCEBEFORE	varchar(255)	255	True

DRMICHAELHUGHSONAFTER	varchar(255)	255	True
DRMICHAELHUGHSONBEFORE	varchar(255)	255	True
DSMBIOLOGICSCOMPANYINCAFTER	varchar(255)	255	True
DSMBIOLOGICSCOMPANYINCBEFORE	varchar(255)	255	True
DUKEHUMANVACCINEINSTITUTEAFTER	varchar(255)	255	True
DUKEHUMANVACCINEINSTITUTEBEFORE	varchar(255)	255	True
DUKEUNIVERSITYAFTER	varchar(255)	255	True
DUKEUNIVERSITYBEFORE	varchar(255)	255	True
EKF-DIAGNOSTICGMBHAFTER	varchar(255)	255	True
EKF-DIAGNOSTICGMBHBEFORE	varchar(255)	255	True
EMAILAFTER	varchar(255)	255	True
EMAILBEFORE	varchar(255)	255	True
EMDSERONOAFTER	varchar(255)	255	True
EMDSERONOBEFORE	varchar(255)	255	True
EMPBIOTECHGMBHAFTER	varchar(255)	255	True
EMPBIOTECHGMBHBEFORE	varchar(255)	255	True
ENGENEICAFTER	varchar(255)	255	True
ENGENEICBEFORE	varchar(255)	255	True
ENZYMATICAABAFTER	varchar(255)	255	True
ENZYMATICAABBEFORE	varchar(255)	255	True
ESBATECHAFTER	varchar(255)	255	True
ESBATECHBEFORE	varchar(255)	255	True
EUROMEDEXAFTER	varchar(255)	255	True
EUROMEDEXBEFORE	varchar(255)	255	True
F-STARBIOTECHNOLOGYLTDAFTER	varchar(255)	255	True
F-STARBIOTECHNOLOGYLTDBEFORE	varchar(255)	255	True
FAXAFTER	varchar(255)	255	True
FAXBEFORE	varchar(255)	255	True
FEGTEXTILTECHNIKAFTER	varchar(255)	255	True
FEGTEXTILTECHNIKBEFORE	varchar(255)	255	True
FISHERSCIENTIFICAFTER	varchar(255)	255	True
FISHERSCIENTIFICBEFORE	varchar(255)	255	True
FISHERSCIENTIFICGMBHAFTER	varchar(255)	255	True
FISHERSCIENTIFICGMBHBEFORE	varchar(255)	255	True
FISHERSCIENTIFICS.A.S.AFTER	varchar(255)	255	True
FISHERSCIENTIFICS.A.S.BEFORE	varchar(255)	255	True
FUJIFILMDIOSYNTHBIOTECHNOLOGIESAFTER	varchar(255)	255	True
FUJIFILMDIOSYNTHBIOTECHNOLOGIESBEFORE	varchar(255)	255	True
FUJIFILMDIOSYNTHBIOTECHNOLOGIESUSAINCAFTER	varchar(255)	255	True
FUJIFILMDIOSYNTHBIOTECHNOLOGIESUSAINCBEFORE	varchar(255)	255	True
GALLUSBIOPHARMAAFTER	varchar(255)	255	True
GALLUSBIOPHARMABEFORE	varchar(255)	255	True
GANYMEDPHARMACEUTICALSAGAFTER	varchar(255)	255	True

GANYMEDPHARMACEUTICALSAGBEFORE	varchar(255)	255	True
GENERONLIMITEDAFTER	varchar(255)	255	True
GENERONLIMITEDBEFORE	varchar(255)	255	True
GENPHARMINTL.,INCAFTER	varchar(255)	255	True
GENPHARMINTL.,INCBEFORE	varchar(255)	255	True
GENPHARMINTLINC.AFTER	varchar(255)	255	True
GENPHARMINTLINC.BEFORE	varchar(255)	255	True
GLAXOSMITHKLINE(CANADA)AFTER	varchar(255)	255	True
GLAXOSMITHKLINE(CANADA)BEFORE	varchar(255)	255	True
GLAXOSMITHKLINE(USA)AFTER	varchar(255)	255	True
GLAXOSMITHKLINE(USA)BEFORE	varchar(255)	255	True
GLAXOSMITHKLINESERVICESUNLIMITEDAFTER	varchar(255)	255	True
GLAXOSMITHKLINESERVICESUNLIMITEDBEFORE	varchar(255)	255	True
GLYCADIAINCAFTER	varchar(255)	255	True
GLYCADIAINCBEFORE	varchar(255)	255	True
GLYKOSFINLANDLIMITEDAFTER	varchar(255)	255	True
GLYKOSFINLANDLIMITEDBEFORE	varchar(255)	255	True
GRIFOLSTHERAPEUTICSINCAFTER	varchar(255)	255	True
GRIFOLSTHERAPEUTICSINCBEFORE	varchar(255)	255	True
GSKAFTER	varchar(255)	255	True
GSKBEFORE	varchar(255)	255	True
GSKSERVICESUNLIMITEDAFTER	varchar(255)	255	True
GSKSERVICESUNLIMITEDBEFORE	varchar(255)	255	True
GULFPHARMACEUTICALINDUSTRIESAFTER	varchar(255)	255	True
GULFPHARMACEUTICALINDUSTRIESBEFORE	varchar(255)	255	True
HAEMONETICSAFTER	varchar(255)	255	True
HAEMONETICSBEFORE	varchar(255)	255	True
HALOZYMETHERAPEUTICSINCAFTER	varchar(255)	255	True
HALOZYMETHERAPEUTICSINCBEFORE	varchar(255)	255	True
HEMARINASAAFTER	varchar(255)	255	True
HEMARINASABEFORE	varchar(255)	255	True
HIGHESTBALANCEAFTER	varchar(255)	255	True
HIGHESTBALANCEBEFORE	varchar(255)	255	True
INSPIRALISAFTER	varchar(255)	255	True
INSPIRALISBEFORE	varchar(255)	255	True
INSTITUTEOFBIOMEDICINEAFTER	varchar(255)	255	True
INSTITUTEOFBIOMEDICINEBEFORE	varchar(255)	255	True
INSTITUTEOFMICROBIALTECHNOLOGYAFTER	varchar(255)	255	True
INSTITUTEOFMICROBIALTECHNOLOGYBEFORE	varchar(255)	255	True
INSTITUTEOFMICROBIOLOGYCASAFTER	varchar(255)	255	True
INSTITUTEOFMICROBIOLOGYCASBEFORE	varchar(255)	255	True
INSTITUTOGRIFOLS,S.A.AFTER	varchar(255)	255	True
INSTITUTOGRIFOLS,S.A.BEFORE	varchar(255)	255	True
INSTITUTOGRIFOLSSAAFTER	varchar(255)	255	True

INSTITUTOGRIFOLSSABEFORE	varchar(255)	255	True
INTERCELLAGAFTER	varchar(255)	255	True
INTERCELLAGBEFORE	varchar(255)	255	True
INTERDISCIPLINARYNANOSCIENCECENTER(INANO)AFTER	varchar(255)	255	True
INTERDISCIPLINARYNANOSCIENCECENTER(INANO)BEFORE	varchar(255)	255	True
INTERNATIONALCENTREFORGENETICENG.ANDBIOTECHAFTER	varchar(255)	255	True
INTERNATIONALCENTREFORGENETICENG.ANDBIOTECHBEFORE	varchar(255)	255	True
INTERNATIONALIMMUNOLOGYCORPORATIONAFTER	varchar(255)	255	True
INTERNATIONALIMMUNOLOGYCORPORATIONBEFORE	varchar(255)	255	True
INTRUMENTATIONLABORATORYAFTER	varchar(255)	255	True
INTRUMENTATIONLABORATORYBEFORE	varchar(255)	255	True
JOUNCETHERAPEUTICS,INCAFTER	varchar(255)	255	True
JOUNCETHERAPEUTICS,INCBEFORE	varchar(255)	255	True
KBIBIOPHARMAINCAFTER	varchar(255)	255	True
KBIBIOPHARMAINCBEFORE	varchar(255)	255	True
KEDRIONSPAAFTER	varchar(255)	255	True
KEDRIONSPABEFORE	varchar(255)	255	True
LAJOLLABIOLOGICSINC.AFTER	varchar(255)	255	True
LAJOLLABIOLOGICSINC.BEFORE	varchar(255)	255	True
LEEBIOSOLUTIONSINCAFTER	varchar(255)	255	True
LEEBIOSOLUTIONSINCBEFORE	varchar(255)	255	True
LEGACYPHARMACEUTICALSSWITZERLANDGMBHAFTER	varchar(255)	255	True
LEGACYPHARMACEUTICALSSWITZERLANDGMBHBEFORE	varchar(255)	255	True
LGLIFESCIENCESLTDHQAFTER	varchar(255)	255	True
LGLIFESCIENCESLTDHQBEFORE	varchar(255)	255	True
LIANYUNGANGHANYARUCHEMICALREAGENTCOLTDAFTER	varchar(255)	255	True
LIANYUNGANGHANYARUCHEMICALREAGENTCOLTDBEFORE	varchar(255)	255	True
LONZABIOLOGICSINCAFTER	varchar(255)	255	True
LONZABIOLOGICSINCBEFORE	varchar(255)	255	True
LONZABIOLOGICSPLCAFTER	varchar(255)	255	True
LONZABIOLOGICSPLCBEFORE	varchar(255)	255	True
MAINESTANDARDSCOMPANYAFTER	varchar(255)	255	True
MAINESTANDARDSCOMPANYBEFORE	varchar(255)	255	True
MAX-PLANCK-INSTITUTFUERAFTER	varchar(255)	255	True
MAX-PLANCK-INSTITUTFUERBEFORE	varchar(255)	255	True
MEDAREX(BMSCOMPANY)AFTER	varchar(255)	255	True
MEDAREX(BMSCOMPANY)BEFORE	varchar(255)	255	True
MEDIMMUNEINCAFTER	varchar(255)	255	True
MEDIMMUNEINCBEFORE	varchar(255)	255	True

MERCKAFTER	varchar(255)	255	True
MERCKBEFORE	varchar(255)	255	True
MERCKSERONOSAAFTER	varchar(255)	255	True
MERCKSERONOSABEFORE	varchar(255)	255	True
MERRIMACKPHARMACEUTICALSAFTER	varchar(255)	255	True
MERRIMACKPHARMACEUTICALSBEFORE	varchar(255)	255	True
MICROMETAGAFTER	varchar(255)	255	True
MICROMETAGBEFORE	varchar(255)	255	True
MILTENYIBIOTECGMBHAFTER	varchar(255)	255	True
MILTENYIBIOTECGMBHBEFORE	varchar(255)	255	True
MINAPHARMPHARMACEUTICALSCHEMICALSINDUSTRIESAFTER	varchar(255)	255	True
MINAPHARMPHARMACEUTICALSCHEMICALSINDUSTRIESBEFORE	varchar(255)	255	True
MODERNATHERAPEUTICSINCAFTER	varchar(255)	255	True
MODERNATHERAPEUTICSINCBEFORE	varchar(255)	255	True
N.VORGANON(MSD)AFTER	varchar(255)	255	True
N.VORGANON(MSD)BEFORE	varchar(255)	255	True
NAMEAFTER	varchar(255)	255	True
NAMEBEFORE	varchar(255)	255	True
NATIONALRESEARCHCOUNCILCANADAAFTER	varchar(255)	255	True
NATIONALRESEARCHCOUNCILCANADABEFORE	varchar(255)	255	True
NCSTATEUNIVERSITYAFTER	varchar(255)	255	True
NCSTATEUNIVERSITYBEFORE	varchar(255)	255	True
NGMBIOPHARMACEUTICALS,INC.AFTER	varchar(255)	255	True
NGMBIOPHARMACEUTICALS,INC.BEFORE	varchar(255)	255	True
NOVARTISAFTER	varchar(255)	255	True
NOVARTISBEFORE	varchar(255)	255	True
NOVAVAX,INCAFTER	varchar(255)	255	True
NOVAVAX,INCBEFORE	varchar(255)	255	True
NOVAVAXINC.AFTER	varchar(255)	255	True
NOVAVAXINC.BEFORE	varchar(255)	255	True
NOVONORDISKASAFTER	varchar(255)	255	True
NOVONORDISKASBEFORE	varchar(255)	255	True
NOVONORDISKPHARMATECHA/SAFTER	varchar(255)	255	True
NOVONORDISKPHARMATECHA/SBEFORE	varchar(255)	255	True
NOVOZYMESBIOPHARMAUKLTDAFTER	varchar(255)	255	True
NOVOZYMESBIOPHARMAUKLTDBEFORE	varchar(255)	255	True
NOVOZYMESSOUTHASIAPVTLTDAFTER	varchar(255)	255	True
NOVOZYMESSOUTHASIAPVTLTDBEFORE	varchar(255)	255	True
NVORGANONAFTER	varchar(255)	255	True
NVORGANONBEFORE	varchar(255)	255	True
OCTAPHARMAAG(AUSTRIA)AFTER	varchar(255)	255	True
OCTAPHARMAAG(AUSTRIA)BEFORE	varchar(255)	255	True
ONCOMEDPHARMACEUTICALSINCAFTER	varchar(255)	255	True

ONCOMEDPHARMACEUTICALSINCBEFORE	varchar(255)	255	True
ORLAPROTEINTECHNOLOGIESLTDAFTER	varchar(255)	255	True
ORLAPROTEINTECHNOLOGIESLTDBEFORE	varchar(255)	255	True
OXFORDUNIVERSITYAFTER	varchar(255)	255	True
OXFORDUNIVERSITYBEFORE	varchar(255)	255	True
PFENEXAFTER	varchar(255)	255	True
PFENEXBEFORE	varchar(255)	255	True
PFIZERINCAFTER	varchar(255)	255	True
PFIZERINCBEFORE	varchar(255)	255	True
PHICOTHERAPEUTICSLTDAFTER	varchar(255)	255	True
PHICOTHERAPEUTICSLTDBEFORE	varchar(255)	255	True
PIERISAGAFTER	varchar(255)	255	True
PIERISAGBEFORE	varchar(255)	255	True
PRICECODEAFTER	varchar(255)	255	True
PRICECODEBEFORE	varchar(255)	255	True
PRIMEPRODUCTSAFTER	varchar(255)	255	True
PRIMEPRODUCTSBEFORE	varchar(255)	255	True
PROBIOGENAGAFTER	varchar(255)	255	True
PROBIOGENAGBEFORE	varchar(255)	255	True
PROMEDIORINCAFTER	varchar(255)	255	True
PROMEDIORINCBEFORE	varchar(255)	255	True
PROMETICBIOPRODUCTIONINCAFTER	varchar(255)	255	True
PROMETICBIOPRODUCTIONINCBEFORE	varchar(255)	255	True
PROMETICBIOTHERAPEUTICSINCAFTER	varchar(255)	255	True
PROMETICBIOTHERAPEUTICSINCBEFORE	varchar(255)	255	True
PROMETICLIFESCIENCESINCAFTER	varchar(255)	255	True
PROMETICLIFESCIENCESINCBEFORE	varchar(255)	255	True
PROTALIXBIOTHERAPEUTICSINCAFTER	varchar(255)	255	True
PROTALIXBIOTHERAPEUTICSINCBEFORE	varchar(255)	255	True
PROTEINARKLTDAFTER	varchar(255)	255	True
PROTEINARKLTDBEFORE	varchar(255)	255	True
PROTEINONEAFTER	varchar(255)	255	True
PROTEINONEBEFORE	varchar(255)	255	True
PROTEINSCIENCESCORPORATIONAFTER	varchar(255)	255	True
PROTEINSCIENCESCORPORATIONBEFORE	varchar(255)	255	True
PROTEOSINCAFTER	varchar(255)	255	True
PROTEOSINCBEFORE	varchar(255)	255	True
PROXCYSBVAFTER	varchar(255)	255	True
PROXCYSBVBEFORE	varchar(255)	255	True
REGENERONPHARMACEUTICALS,INC.AFTER	varchar(255)	255	True
REGENERONPHARMACEUTICALS,INC.BEFORE	varchar(255)	255	True
RELIANCELIFESCIENCESPVTLTDAFTER	varchar(255)	255	True
RELIANCELIFESCIENCESPVTLTDBEFORE	varchar(255)	255	True
RENSSELEARPOLYTECHNICINSTITUTEAFTER	varchar(255)	255	True

RENSSELEARPOLYTECHNICINSTITUTEBEFORE	varchar(255)	255	True
REPLIGENAFTER	varchar(255)	255	True
REPLIGENBEFORE	varchar(255)	255	True
RICHORELIFESCIENCESPVTLTDAFTER	varchar(255)	255	True
RICHORELIFESCIENCESPVTLTDBEFORE	varchar(255)	255	True
RICHTER-HELMBIOLOGICSGMBH&CO.KGAFTER	varchar(255)	255	True
RICHTER-HELMBIOLOGICSGMBH&CO.KGBEFORE	varchar(255)	255	True
RUTGERSUNIVERSITYAFTER	varchar(255)	255	True
RUTGERSUNIVERSITYBEFORE	varchar(255)	255	True
RYANSCIENTIFICINC.AFTER	varchar(255)	255	True
RYANSCIENTIFICINC.BEFORE	varchar(255)	255	True
SAESCIENTIFICAAFTER	varchar(255)	255	True
SAESCIENTIFICABEFORE	varchar(255)	255	True
SANOFI-AVENTISAFTER	varchar(255)	255	True
SANOFI-AVENTISBEFORE	varchar(255)	255	True
SANTENKITRADINGSINCAFTER	varchar(255)	255	True
SANTENKITRADINGSINCBEFORE	varchar(255)	255	True
SCILPROTEINSGMBHAFTER	varchar(255)	255	True
SCILPROTEINSGMBHBEFORE	varchar(255)	255	True
SCRUMINCAFTER	varchar(255)	255	True
SCRUMINCBEFORE	varchar(255)	255	True
SERCOAFTER	varchar(255)	255	True
SERCOBEFORE	varchar(255)	255	True
SHASUNPHARMACEUTICALSLTDAFTER	varchar(255)	255	True
SHASUNPHARMACEUTICALSLTDBEFORE	varchar(255)	255	True
SHENZHENHEP0ALINKPHARMACEUTICALSCO.LTDAFTER	varchar(255)	255	True
SHENZHENHEP0ALINKPHARMACEUTICALSCO.LTDBEFORE	varchar(255)	255	True
SHIPPINGINSTRSAFTER	varchar(255)	255	True
SHIPPINGINSTRSBEFORE	varchar(255)	255	True
SHIPPOSTALCODEAFTER	varchar(255)	255	True
SHIPPOSTALCODEBEFORE	varchar(255)	255	True
SHIPTOADDR1AFTER	varchar(255)	255	True
SHIPTOADDR1BEFORE	varchar(255)	255	True
SHIPTOADDR2AFTER	varchar(255)	255	True
SHIPTOADDR2BEFORE	varchar(255)	255	True
SHIPTOADDR3AFTER	varchar(255)	255	True
SHIPTOADDR3BEFORE	varchar(255)	255	True
SHIPTOADDR3LOCAFTER	varchar(255)	255	True
SHIPTOADDR3LOCBEFORE	varchar(255)	255	True
SHIPTOADDR4AFTER	varchar(255)	255	True
SHIPTOADDR4BEFORE	varchar(255)	255	True
SHIPTOADDR5AFTER	varchar(255)	255	True
SHIPTOADDR5BEFORE	varchar(255)	255	True

SHORTNAMEAFTER	varchar(255)	255	True
SHORTNAMEBEFORE	varchar(255)	255	True
SIAMED'XPRESSAFTER	varchar(255)	255	True
SIAMED'XPRESSBEFORE	varchar(255)	255	True
SIGMA-ALDRICHCORPORATIONAFTER	varchar(255)	255	True
SIGMA-ALDRICHCORPORATIONBEFORE	varchar(255)	255	True
SIGNALPHARMACEUTICALSLLCAFTER	varchar(255)	255	True
SIGNALPHARMACEUTICALSLLCBEFORE	varchar(255)	255	True
SINOPHARMCHEMICALREAGENTCO.LTDAFTER	varchar(255)	255	True
SINOPHARMCHEMICALREAGENTCO.LTDBEFORE	varchar(255)	255	True
SIREANALYTICALSYSTEMSAFTER	varchar(255)	255	True
SIREANALYTICALSYSTEMSBEFORE	varchar(255)	255	True
SODEFAULTTYPEAFTER	varchar(255)	255	True
SODEFAULTTYPEBEFORE	varchar(255)	255	True
SOLDPOSTALCODEAFTER	varchar(255)	255	True
SOLDPOSTALCODEBEFORE	varchar(255)	255	True
SOLDTOADDR1AFTER	varchar(255)	255	True
SOLDTOADDR1BEFORE	varchar(255)	255	True
SOLDTOADDR2AFTER	varchar(255)	255	True
SOLDTOADDR2BEFORE	varchar(255)	255	True
SOLDTOADDR3AFTER	varchar(255)	255	True
SOLDTOADDR3BEFORE	varchar(255)	255	True
SOLDTOADDR3LOCAFTER	varchar(255)	255	True
SOLDTOADDR3LOCBEFORE	varchar(255)	255	True
SOLDTOADDR4AFTER	varchar(255)	255	True
SOLDTOADDR4BEFORE	varchar(255)	255	True
SOLDTOADDR5AFTER	varchar(255)	255	True
SOLDTOADDR5BEFORE	varchar(255)	255	True
SUNPHARMACEUTICALINDUSTRIESLTDAFTER	varchar(255)	255	True
SUNPHARMACEUTICALINDUSTRIESLTDBEFORE	varchar(255)	255	True
SUNYEARSCIENTIFICINCAFTER	varchar(255)	255	True
SUNYEARSCIENTIFICINCBEFORE	varchar(255)	255	True
SUTROBIOPHARMAAFTER	varchar(255)	255	True
SUTROBIOPHARMABEFORE	varchar(255)	255	True
SYMMETRIXBIOTECHPVT,LTD.AFTER	varchar(255)	255	True
SYMMETRIXBIOTECHPVT,LTD.BEFORE	varchar(255)	255	True
SYNGENEINTERNATIONALLTDAFTER	varchar(255)	255	True
SYNGENEINTERNATIONALLTDBEFORE	varchar(255)	255	True
TAXEXEMPTNUMBERAFTER	varchar(255)	255	True
TAXEXEMPTNUMBERBEFORE	varchar(255)	255	True
TAXSTATUSAFTER	varchar(255)	255	True
TAXSTATUSBEFORE	varchar(255)	255	True
TELEPHONEAFTER	varchar(255)	255	True
TELEPHONEBEFORE	varchar(255)	255	True

TEVABIOPHARMACEUTICALSUSAINC.AFTER	varchar(255)	255	True
TEVABIOPHARMACEUTICALSUSAINC.BEFORE	varchar(255)	255	True
THERMOFISHERSCIENTIFICAFTER	varchar(255)	255	True
THERMOFISHERSCIENTIFICBEFORE	varchar(255)	255	True
THEUNIVERSITYOFMANCHESTERAFTER	varchar(255)	255	True
THEUNIVERSITYOFMANCHESTERBEFORE	varchar(255)	255	True
TNOEARTH,ENVIRONMENTAL&LSTNOAFTER	varchar(255)	255	True
TNOEARTH,ENVIRONMENTAL&LSTNOBEFORE	varchar(255)	255	True
TRANSACTIONNATUREAFTER	varchar(255)	255	True
TRANSACTIONNATUREBEFORE	varchar(255)	255	True
TRANSACTIONNATURECAFTER	varchar(255)	255	True
TRANSACTIONNATURECBEFORE	varchar(255)	255	True
TROPHOGENAFTER	varchar(255)	255	True
TROPHOGENBEFORE	varchar(255)	255	True
TUMUNCHENAFTER	varchar(255)	255	True
TUMUNCHENBEFORE	varchar(255)	255	True
UCBCELLTECHAFTER	varchar(255)	255	True
UCBCELLTECHBEFORE	varchar(255)	255	True
UKVATFLAGAFTER	varchar(255)	255	True
UKVATFLAGBEFORE	varchar(255)	255	True
UNIVERSITYCOLLEGELONDONAFTER	varchar(255)	255	True
UNIVERSITYCOLLEGELONDONBEFORE	varchar(255)	255	True
UNIVERSITYOFBRITISHCOLUMBIAAFTER	varchar(255)	255	True
UNIVERSITYOFBRITISHCOLUMBIABEFORE	varchar(255)	255	True
UNIVERSITYOFCOPENHAGENAFTER	varchar(255)	255	True
UNIVERSITYOFCOPENHAGENBEFORE	varchar(255)	255	True
UNIVERSITYOFGDANSKAFTER	varchar(255)	255	True
UNIVERSITYOFGDANSKBEFORE	varchar(255)	255	True
UNIVERSITYOFHYDERABADAFTER	varchar(255)	255	True
UNIVERSITYOFHYDERABADBEFORE	varchar(255)	255	True
UNIVERSITYOFKENTUCKYAFTER	varchar(255)	255	True
UNIVERSITYOFKENTUCKYBEFORE	varchar(255)	255	True
UNIVERSITYOFZURICHAFTER	varchar(255)	255	True
UNIVERSITYOFZURICHBEFORE	varchar(255)	255	True
VERAXBIOMEDICALAFTER	varchar(255)	255	True
VERAXBIOMEDICALBEFORE	varchar(255)	255	True
VITACYTELLCAFTER	varchar(255)	255	True
VITACYTELLCBEFORE	varchar(255)	255	True
VIVOSTATA/SAFTER	varchar(255)	255	True
VIVOSTATA/SBEFORE	varchar(255)	255	True
VWRINTERNATIONALAFTER	varchar(255)	255	True
VWRINTERNATIONALBEFORE	varchar(255)	255	True
VWRINTERNATIONALBVBAAFTER	varchar(255)	255	True
VWRINTERNATIONALBVBABEFORE	varchar(255)	255	True

WAKOCHEMICALSGMBHAFTER	varchar(255)	255	True
WAKOCHEMICALSGMBHBEFORE	varchar(255)	255	True
WEB-LABLABORATORYPRODUCTSCORPAFTER	varchar(255)	255	True
WEB-LABLABORATORYPRODUCTSCORPBEFORE	varchar(255)	255	True
WOOILSCIENCESAFTER	varchar(255)	255	True
WOOILSCIENCESBEFORE	varchar(255)	255	True
YALEUNIVERSITYBCMMAFTER	varchar(255)	255	True
YALEUNIVERSITYBCMMBEFORE	varchar(255)	255	True
YICHANGCHANGJIANGPHARMACEUTICALLTDAFTER	varchar(255)	255	True
YICHANGCHANGJIANGPHARMACEUTICALLTDBEFORE	varchar(255)	255	True
YORKSHIREPROCESSTECHNOLOGYAFTER	varchar(255)	255	True
YORKSHIREPROCESSTECHNOLOGYBEFORE	varchar(255)	255	True
ZYMETECHEHFAFTER	varchar(255)	255	True
ZYMETECHEHFBEFORE	varchar(255)	255	True
ZYMEWORKSAFTER	varchar(255)	255	True
ZYMEWORKSBEFORE	varchar(255)	255	True
ZYMOGENETICSINCAFTER	varchar(255)	255	True
ZYMOGENETICSINCBEFORE	varchar(255)	255	True
ZYNGENIAAFTER	varchar(255)	255	True
ZYNGENIABEFORE	varchar(255)	255	True
DELTAPRECISIONLTDAFTER	varchar(255)	255	True
DELTAPRECISIONLTDBEFORE	varchar(255)	255	True
KRISHGENBIOSYSTEMSAFTER	varchar(255)	255	True
KRISHGENBIOSYSTEMSBEFORE	varchar(255)	255	True
LFBBIOTECHNOLOGIESAFTER	varchar(255)	255	True
LFBBIOTECHNOLOGIESBEFORE	varchar(255)	255	True
MIKEBANUAAFTER	varchar(255)	255	True
MIKEBANUABEFORE	varchar(255)	255	True
MILLIPORESIGMAAFTER	varchar(255)	255	True
MILLIPORESIGMABEFORE	varchar(255)	255	True
PHILOGENS.P.AAFTER	varchar(255)	255	True
PHILOGENS.P.ABEFORE	varchar(255)	255	True
PNUVAXINCORPORATEDAFTER	varchar(255)	255	True
PNUVAXINCORPORATEDBEFORE	varchar(255)	255	True
TEVAPHARMACEUTICALINDUSTRIESLTDAFTER	varchar(255)	255	True
TEVAPHARMACEUTICALINDUSTRIESLTDBEFORE	varchar(255)	255	True
UNIVERSITÉLAVALAFTER	varchar(255)	255	True
UNIVERSITÉLAVALBEFORE	varchar(255)	255	True
VWRINTERNATIONALGMBHAFTER	varchar(255)	255	True
VWRINTERNATIONALGMBHBEFORE	varchar(255)	255	True
WHPARTNERSHIPLIMITEDAFTER	varchar(255)	255	True
WHPARTNERSHIPLIMITEDBEFORE	varchar(255)	255	True
YICHANGHECCHANGJIANGPHARMACEUTICALCO.LTD AFTER	varchar(255)	255	True

YICHANGHECCHANGJIANGPHARMACEUTICALCO.LTD BEFORE	varchar(255)	255	True
YICHANGHECCHANGJIANGPHARMACEUTICALCOLTDAFTER	varchar(255)	255	True
YICHANGHECCHANGJIANGPHARMACEUTICALCOLTDBEFORE	varchar(255)	255	True
CELLTRICKSBIOTECHPVTLTDAFTER	varchar(255)	255	True
CELLTRICKSBIOTECHPVTLTDBEFORE	varchar(255)	255	True
COOKREGENTECAFTER	varchar(255)	255	True
COOKREGENTECBEFORE	varchar(255)	255	True
EPYGENBIOTECHAFTER	varchar(255)	255	True
EPYGENBIOTECHBEFORE	varchar(255)	255	True
GREENCROSSCORPORATIONAFTER	varchar(255)	255	True
GREENCROSSCORPORATIONBEFORE	varchar(255)	255	True
INSTITUTOSUPERIORTECNICO(IST-ID)AFTER	varchar(255)	255	True
INSTITUTOSUPERIORTECNICO(IST-ID)BEFORE	varchar(255)	255	True
LEEHYOBIOSCIENCECO.,LTD.AFTER	varchar(255)	255	True
LEEHYOBIOSCIENCECO.,LTD.BEFORE	varchar(255)	255	True
ZEPTEONAFTER	varchar(255)	255	True
ZEPTEONBEFORE	varchar(255)	255	True
TESTAFTER	varchar(255)	255	True
TESTBEFORE	varchar(255)	255	True
HOCHSCHULEMANNHEIMINSTITUTFÜRBIOCHEMIEAFTER	varchar(255)	255	True
HOCHSCHULEMANNHEIMINSTITUTFÜRBIOCHEMIEBEFORE	varchar(255)	255	True
INTERPHARMBIOTEKLTDAFTER	varchar(255)	255	True
INTERPHARMBIOTEKLTDBEFORE	varchar(255)	255	True

SQL Script

```
CREATE TABLE [History].[ArchiveArCustomer27OCT2016112011030]
(
[TransactionDescription] [varchar] (150) COLLATE Latin1_General_BIN NULL,
[DatabaseName] [varchar] (150) COLLATE Latin1_General_BIN NOT NULL,
[SignatureDateTime] [datetime2] NOT NULL,
[Operator] [varchar] (20) COLLATE Latin1_General_BIN NOT NULL,
[ItemKey] [varchar] (150) COLLATE Latin1_General_BIN NOT NULL,
[ComputerName] [varchar] (150) COLLATE Latin1_General_BIN NULL,
[ProgramName] [varchar] (100) COLLATE Latin1_General_BIN NOT NULL,
[ConditionName] [varchar] (15) COLLATE Latin1_General_BIN NULL,
[AlreadyEntered] [bit] NULL,
[A2STECHNOLOGIESAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[A2STECHNOLOGIESBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[AALTOSCIENTIFICLIMITEDAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[AALTOSCIENTIFICLIMITEDBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[ABBOMAXINC.AFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[ABBOMAXINC.BEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[ABBOTTLABORATORIESAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[ABBOTTLABORATORIESBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
```

Author: Johnson, Chris

```
[ABBOTTPOINTOFCARECANADALTDAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[ABBOTTPOINTOFCARECANADALTDBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[ABBVIEBIORESEARCHCENTERINCAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[ABBVIEBIORESEARCHCENTERINCBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[ABSCILLCAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[ABSCILLCBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[ABSOLUTEANTIBODYAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[ABSOLUTEANTIBODYBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[ADBIOTECHSOLUTIONSAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[ADBIOTECHSOLUTIONSBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[ADCBIOTECHNOLOGYAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[ADCBIOTECHNOLOGYBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[ADNEXUS-ABMSR&DCOMPANY(0590)AFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[ADNEXUS-ABMSR&DCOMPANY(0590)BEFORE] [varchar] (255) COLLATE Latin1_General_BIN
NULL,
[ADVYLTDAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[ADVYLTDBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[AGGAMINBIOLOGICSAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[AGGAMINBIOLOGICSBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[AJUSTHOLDINGA/SAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[AJUSTHOLDINGA/SBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[ALLERGANINC.AFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[ALLERGANINC.BEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[ALLERGANSALESLLCAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[ALLERGANSALESLLCBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[ALTMETHODFLAGAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[ALTMETHODFLAGBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[AMBRX,INC.AFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[AMBRX,INC.BEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[AMGENAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[AMGENBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[AMGENRESEARCH(MUNICH)GMBHAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[AMGENRESEARCH(MUNICH)GMBHBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[ANASPECINCAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[ANASPECINCBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[AREAAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[AREABEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[ARKRAYAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[ARKRAYBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[ASPENPHARMACAREAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[ASPENPHARMACAREBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[ATOLLGMBHAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[ATOLLGMBHBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[AUXILIUMPHARMACEUTICALSINCAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[AUXILIUMPHARMACEUTICALSINCBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[AVAXIABIOLOGICSINCAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[AVAXIABIOLOGICSINCBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[AVIDBIOSERVICESAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[AVIDBIOSERVICESBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[AVITIDE,INC.AFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[AVITIDE,INC.BEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[B-BRIDGEINTERNATIONALINCAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[B-BRIDGEINTERNATIONALINCBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[BACB.V.AFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[BACB.V.BEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[BAYLORCOLLEGEOFMEDICINEAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[BAYLORCOLLEGEOFMEDICINEBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[BEIJINGZHONGYUANLIMITEDAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
```

```
[BEIJINGZHONGYUANLIMITEDBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[BIO-RADABDSEROTECGMBHAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[BIO-RADABDSEROTECGMBHBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[BIO-RADDIAGNOSTICSINCAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[BIO-RADDIAGNOSTICSINCBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[BIO-RADLABORATORIESDEESIDELTDAFTER] [varchar] (255) COLLATE Latin1_General_BIN
NULL,
[BIO-RADLABORATORIESDEESIDELTDBEFORE] [varchar] (255) COLLATE Latin1_General_BIN
NULL,
[BIO-RADLABORATORIESINCAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[BIO-RADLABORATORIESINCBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[BIO-TECHNOLOGYGENERAL(ISRAEL)LTDAFTER] [varchar] (255) COLLATE Latin1_General_BIN
NULL,
[BIO-TECHNOLOGYGENERAL(ISRAEL)LTDBEFORE] [varchar] (255) COLLATE Latin1_General_BIN
NULL,
[BIOANALYTICMACEIJSTOPAAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[BIOANALYTICMACEIJSTOPABEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[BIOCONRESEARCHLIMITEDSEZUNITAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[BIOCONRESEARCHLIMITEDSEZUNITBEFORE] [varchar] (255) COLLATE Latin1_General_BIN
NULL,
[BIOINVENTINTERNATIONALABAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[BIOINVENTINTERNATIONALABBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[BIONISYSRDCORPAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[BIONISYSRDCORPBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[BIORADAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[BIORADBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[BIORADLABORATORIESAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[BIORADLABORATORIESBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[BIORESOURCETECHNOLOGYAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[BIORESOURCETECHNOLOGYBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[BIOTESTAGAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[BIOTESTAGBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[BIOTRADERINCAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[BIOTRADERINCBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[BOEHRINGERINGELHEIMAUSTRIAGMBHAFTER] [varchar] (255) COLLATE Latin1_General_BIN
NULL,
[BOEHRINGERINGELHEIMAUSTRIAGMBHBEFORE] [varchar] (255) COLLATE Latin1_General_BIN
NULL,
[BRANCHAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[BRANCHBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[BRISTOL-MYERSSQUIBBCOAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[BRISTOL-MYERSSQUIBBCOBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[BRISTOL-MYERSSQUIBBHOPKINTONAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[BRISTOL-MYERSSQUIBBHOPKINTONBEFORE] [varchar] (255) COLLATE Latin1_General_BIN
NULL,
[BRITISHCOLUMBIAUNIVERSITYAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[BRITISHCOLUMBIAUNIVERSITYBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[CADILAPHARMACEUTICALSLTDAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[CADILAPHARMACEUTICALSLTDBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[CATALENTPHARMASOLUTIONS,LLCAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[CATALENTPHARMASOLUTIONS,LLCBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[CATALENTPHARMASOLUTIONSAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[CATALENTPHARMASOLUTIONSBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[CEDARLANELABORATORIESUSAAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[CEDARLANELABORATORIESUSABEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[CELSISINTERNATIONALGMBHAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[CELSISINTERNATIONALGMBHBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[CEPHALONAUSTRALIAAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[CEPHALONAUSTRALIABEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
```

```
[CHEMDIVINCAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[CHEMDIVINCBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[CHEMQUESTLIMITEDAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[CHEMQUESTLIMITEDBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[CINTRADEENTERPRISE,INC.AFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[CINTRADEENTERPRISE,INC.BEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[CJSCGENERIUMAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[CJSCGENERIUMBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[COBRABIOLOGICSLTDAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[COBRABIOLOGICSLTDBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[COMPANYTAXNUMBERAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[COMPANYTAXNUMBERBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[CONTACTAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[CONTACTBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[CONTRACTPRCREQDAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[CONTRACTPRCREQDBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[COOKPHARMICAAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[COOKPHARMICABEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[CRESCENDOBIOLOGICSLTDAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[CRESCENDOBIOLOGICSLTDBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[CRUCELLBIOLOGICSINCC/OJOHNSON&JOHNSONAFTER] [varchar] (255) COLLATE Latin1_General_-
BIN NULL,
[CRUCELLBIOLOGICSINCC/OJOHNSON&JOHNSONBEFORE] [varchar] (255) COLLATE Latin1_-
General_BIN NULL,
[CRUCELLHOLLANDBVAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[CRUCELLHOLLANDBVBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[CSLBEHRINGAGAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[CSLBEHRINGAGBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[CSLBEHRINGGMBHAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[CSLBEHRINGGMBHBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[CSLLTDAUSTRALIAAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[CSLLTDAUSTRALIABEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[CURRENCYAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[CURRENCYBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[CYPRESSINTERNATIONALLTDAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[CYPRESSINTERNATIONALLTDBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[D.B.AITALIASRLAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[D.B.AITALIASRLBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[DELIVERYTERMSAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[DELIVERYTERMSBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[DISCIENCEAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[DISCIENCEBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[DRMICHAELHUGHSONAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[DRMICHAELHUGHSONBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[DSMBIOLOGICSCOMPANYINCAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[DSMBIOLOGICSCOMPANYINCBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[DUKEHUMANVACCINEINSTITUTEAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[DUKEHUMANVACCINEINSTITUTEBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[DUKEUNIVERSITYAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[DUKEUNIVERSITYBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[EKF-DIAGNOSTICGMBHAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[EKF-DIAGNOSTICGMBHBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[EMAILAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[EMAILBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[EMDSERONOAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[EMDSERONOBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[EMPBIOTECHGMBHAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[EMPBIOTECHGMBHBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
```

```
[ENGENEICAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[ENGENEICBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[ENZYMATICAABAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[ENZYMATICAABBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[ESBATECHAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[ESBATECHBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[EUROMEDEXAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[EUROMEDEXBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[F-STARBIOTECHNOLOGYLTDAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[F-STARBIOTECHNOLOGYLTDBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[FAXAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[FAXBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[FEGTEXTILTECHNIKAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[FEGTEXTILTECHNIKBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[FISHERSCIENTIFICAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[FISHERSCIENTIFICBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[FISHERSCIENTIFICGMBHAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[FISHERSCIENTIFICGMBHBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[FISHERSCIENTIFICS.A.S.AFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[FISHERSCIENTIFICS.A.S.BEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[FUJIFILMDIOSYNTHBIOTECHNOLOGIESAFTER] [varchar] (255) COLLATE Latin1_General_BIN
NULL,
[FUJIFILMDIOSYNTHBIOTECHNOLOGIESBEFORE] [varchar] (255) COLLATE Latin1_General_BIN
NULL,
[FUJIFILMDIOSYNTHBIOTECHNOLOGIESUSAINCAFTER] [varchar] (255) COLLATE Latin1_General_-
BIN NULL,
[FUJIFILMDIOSYNTHBIOTECHNOLOGIESUSAINCBEFORE] [varchar] (255) COLLATE Latin1_-
General_BIN NULL,
[GALLUSBIOPHARMAAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[GALLUSBIOPHARMABEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[GANYMEDPHARMACEUTICALSAGAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[GANYMEDPHARMACEUTICALSAGBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[GENERONLIMITEDAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[GENERONLIMITEDBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[GENPHARMINTL.,INCAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[GENPHARMINTL.,INCBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[GENPHARMINTLINC.AFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[GENPHARMINTLINC.BEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[GLAXOSMITHKLINE(CANADA)AFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[GLAXOSMITHKLINE(CANADA)BEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[GLAXOSMITHKLINE(USA)AFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[GLAXOSMITHKLINE(USA)BEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[GLAXOSMITHKLINESERVICESUNLIMITEDAFTER] [varchar] (255) COLLATE Latin1_General_BIN
NULL,
[GLAXOSMITHKLINESERVICESUNLIMITEDBEFORE] [varchar] (255) COLLATE Latin1_General_BIN
NULL,
[GLYCADIAINCAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[GLYCADIAINCBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[GLYKOSFINLANDLIMITEDAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[GLYKOSFINLANDLIMITEDBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[GRIFOLSTHERAPEUTICSINCAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[GRIFOLSTHERAPEUTICSINCBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[GSKAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[GSKBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[GSKSERVICESUNLIMITEDAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[GSKSERVICESUNLIMITEDBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[GULFPHARMACEUTICALINDUSTRIESAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[GULFPHARMACEUTICALINDUSTRIESBEFORE] [varchar] (255) COLLATE Latin1_General_BIN
NULL,
```

Author: Johnson, Chris

```
[HAEMONETICSAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[HAEMONETICSBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[HALOZYMETHERAPEUTICSINCAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[HALOZYMETHERAPEUTICSINCBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[HEMARINASAAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[HEMARINASABEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[HIGHESTBALANCEAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[HIGHESTBALANCEBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[INSPIRALISAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[INSPIRALISBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[INSTITUTEOFBIOMEDICINEAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[INSTITUTEOFBIOMEDICINEBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[INSTITUTEOFMICROBIALTECHNOLOGYAFTER] [varchar] (255) COLLATE Latin1_General_BIN
NULL,
[INSTITUTEOFMICROBIALTECHNOLOGYBEFORE] [varchar] (255) COLLATE Latin1_General_BIN
NULL,
[INSTITUTEOFMICROBIOLOGYCASAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[INSTITUTEOFMICROBIOLOGYCASBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[INSTITUTOGRIFOLS,S.A.AFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[INSTITUTOGRIFOLS,S.A.BEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[INSTITUTOGRIFOLSSAAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[INSTITUTOGRIFOLSSABEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[INTERCELLAGAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[INTERCELLAGBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[INTERDISCIPLINARYNANOSCIENCECENTER(INANO)AFTER] [varchar] (255) COLLATE Latin1_-
General_BIN NULL,
[INTERDISCIPLINARYNANOSCIENCECENTER(INANO)BEFORE] [varchar] (255) COLLATE Latin1_-
General_BIN NULL,
[INTERNATIONALCENTREFORGENETICENG.ANDBIOTECHAFTER] [varchar] (255) COLLATE Latin1_-
General_BIN NULL,
[INTERNATIONALCENTREFORGENETICENG.ANDBIOTECHBEFORE] [varchar] (255) COLLATE Latin1_-
General_BIN NULL,
[INTERNATIONALIMMUNOLOGYCORPORATIONAFTER] [varchar] (255) COLLATE Latin1_General_BIN
NULL,
[INTERNATIONALIMMUNOLOGYCORPORATIONBEFORE] [varchar] (255) COLLATE Latin1_General_-
BIN NULL,
[INTRUMENTATIONLABORATORYAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[INTRUMENTATIONLABORATORYBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[JOUNCETHERAPEUTICS,INCAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[JOUNCETHERAPEUTICS,INCBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[KBIBIOPHARMAINCAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[KBIBIOPHARMAINCBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[KEDRIONSPAAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[KEDRIONSPABEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[LAJOLLABIOLOGICSINC.AFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[LAJOLLABIOLOGICSINC.BEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[LEEBIOSOLUTIONSINCAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[LEEBIOSOLUTIONSINCBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[LEGACYPHARMACEUTICALSSWITZERLANDGMBHAFTER] [varchar] (255) COLLATE Latin1_General_-
BIN NULL,
[LEGACYPHARMACEUTICALSSWITZERLANDGMBHBEFORE] [varchar] (255) COLLATE Latin1_General_-
BIN NULL,
[LGLIFESCIENCESLTDHQAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[LGLIFESCIENCESLTDHQBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[LIANYUNGANGHANYARUCHEMICALREAGENTCOLTDAFTER] [varchar] (255) COLLATE Latin1_-
General_BIN NULL,
[LIANYUNGANGHANYARUCHEMICALREAGENTCOLTDBEFORE] [varchar] (255) COLLATE Latin1_-
General_BIN NULL,
[LONZABIOLOGICSINCAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[LONZABIOLOGICSINCBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
```

```
[LONZABIOLOGICSPLCAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[LONZABIOLOGICSPLCBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[MAINESTANDARDSCOMPANYAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[MAINESTANDARDSCOMPANYBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[MAX-PLANCK-INSTITUTFUERAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[MAX-PLANCK-INSTITUTFUERBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[MEDAREX(BMSCOMPANY)AFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[MEDAREX(BMSCOMPANY)BEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[MEDIMMUNEINCAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[MEDIMMUNEINCBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[MERCKAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[MERCKBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[MERCKSERONOSAAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[MERCKSERONOSABEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[MERRIMACKPHARMACEUTICALSAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[MERRIMACKPHARMACEUTICALSBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[MICROMETAGAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[MICROMETAGBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[MILTENYIBIOTECGMBHAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[MILTENYIBIOTECGMBHBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[MINAPHARMPHARMACEUTICALSCHEMICALSINDUSTRIESAFTER] [varchar] (255) COLLATE Latin1_-
General_BIN NULL,
[MINAPHARMPHARMACEUTICALSCHEMICALSINDUSTRIESBEFORE] [varchar] (255) COLLATE Latin1_-
General_BIN NULL,
[MODERNATHERAPEUTICSINCAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[MODERNATHERAPEUTICSINCBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[N.VORGANON(MSD)AFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[N.VORGANON(MSD)BEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[NAMEAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[NAMEBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[NATIONALRESEARCHCOUNCILCANADAAFTER] [varchar] (255) COLLATE Latin1_General_BIN
NULL,
[NATIONALRESEARCHCOUNCILCANADABEFORE] [varchar] (255) COLLATE Latin1_General_BIN
NULL,
[NCSTATEUNIVERSITYAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[NCSTATEUNIVERSITYBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[NGMBIOPHARMACEUTICALS,INC.AFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[NGMBIOPHARMACEUTICALS,INC.BEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[NOVARTISAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[NOVARTISBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[NOVAVAX,INCAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[NOVAVAX,INCBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[NOVAVAXINC.AFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[NOVAVAXINC.BEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[NOVONORDISKASAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[NOVONORDISKASBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[NOVONORDISKPHARMATECHA/SAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[NOVONORDISKPHARMATECHA/SBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[NOVOZYMESBIOPHARMAUKLTDAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[NOVOZYMESBIOPHARMAUKLTDBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[NOVOZYMESSOUTHASIAPVTLTDAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[NOVOZYMESSOUTHASIAPVTLTDBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[NVORGANONAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[NVORGANONBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[OCTAPHARMAAG(AUSTRIA)AFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[OCTAPHARMAAG(AUSTRIA)BEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[ONCOMEDPHARMACEUTICALSINCAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[ONCOMEDPHARMACEUTICALSINCBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
```

Author: Johnson, Chris

```
[ORLAPROTEINTECHNOLOGIESLTDAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[ORLAPROTEINTECHNOLOGIESLTDBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[OXFORDUNIVERSITYAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[OXFORDUNIVERSITYBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[PFENEXAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[PFENEXBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[PFIZERINCAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[PFIZERINCBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[PHICOTHERAPEUTICSLTDAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[PHICOTHERAPEUTICSLTDBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[PIERISAGAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[PIERISAGBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[PRICECODEAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[PRICECODEBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[PRIMEPRODUCTSAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[PRIMEPRODUCTSBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[PROBIOGENAGAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[PROBIOGENAGBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[PROMEDIORINCAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[PROMEDIORINCBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[PROMETICBIOPRODUCTIONINCAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[PROMETICBIOPRODUCTIONINCBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[PROMETICBIOTHERAPEUTICSINCAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[PROMETICBIOTHERAPEUTICSINCBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[PROMETICLIFESCIENCESINCAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[PROMETICLIFESCIENCESINCBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[PROTALIXBIOTHERAPEUTICSINCAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[PROTALIXBIOTHERAPEUTICSINCBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[PROTEINARKLTDAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[PROTEINARKLTDBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[PROTEINONEAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[PROTEINONEBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[PROTEINSCIENCESCORPORATIONAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[PROTEINSCIENCESCORPORATIONBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[PROTEOSINCAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[PROTEOSINCBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[PROXCYSBVAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[PROXCYSBVBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[REGENERONPHARMACEUTICALS,INC.AFTER] [varchar] (255) COLLATE Latin1_General_BIN
NULL,
[REGENERONPHARMACEUTICALS,INC.BEFORE] [varchar] (255) COLLATE Latin1_General_BIN
NULL,
[RELIANCELIFESCIENCESPVTLTDAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[RELIANCELIFESCIENCESPVTLTDBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[RENSSELEARPOLYTECHNICINSTITUTEAFTER] [varchar] (255) COLLATE Latin1_General_BIN
NULL,
[RENSSELEARPOLYTECHNICINSTITUTEBEFORE] [varchar] (255) COLLATE Latin1_General_BIN
NULL,
[REPLIGENAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[REPLIGENBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[RICHORELIFESCIENCESPVTLTDAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[RICHORELIFESCIENCESPVTLTDBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[RICHTER-HELMBIOLOGICSGMBH&CO.KGAFTER] [varchar] (255) COLLATE Latin1_General_BIN
NULL,
[RICHTER-HELMBIOLOGICSGMBH&CO.KGBEFORE] [varchar] (255) COLLATE Latin1_General_BIN
NULL,
[RUTGERSUNIVERSITYAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[RUTGERSUNIVERSITYBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[RYANSCIENTIFICINC.AFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
```

```
[RYANSCIENTIFICINC.BEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[SAESCIENTIFICAAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[SAESCIENTIFICABEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[SANOFI-AVENTISAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[SANOFI-AVENTISBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[SANTENKITRADINGSINCAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[SANTENKITRADINGSINCBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[SCILPROTEINSGMBHAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[SCILPROTEINSGMBHBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[SCRUMINCAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[SCRUMINCBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[SERCOAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[SERCOBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[SHASUNPHARMACEUTICALSLTDAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[SHASUNPHARMACEUTICALSLTDBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[SHENZHENHEP0ALINKPHARMACEUTICALSCO.LTDAFTER] [varchar] (255) COLLATE Latin1_-
General_BIN NULL,
[SHENZHENHEP0ALINKPHARMACEUTICALSCO.LTDBEFORE] [varchar] (255) COLLATE Latin1_-
General_BIN NULL,
[SHIPPINGINSTRSAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[SHIPPINGINSTRSBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[SHIPPOSTALCODEAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[SHIPPOSTALCODEBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[SHIPTOADDR1AFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[SHIPTOADDR1BEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[SHIPTOADDR2AFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[SHIPTOADDR2BEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[SHIPTOADDR3AFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[SHIPTOADDR3BEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[SHIPTOADDR3LOCAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[SHIPTOADDR3LOCBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[SHIPTOADDR4AFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[SHIPTOADDR4BEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[SHIPTOADDR5AFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[SHIPTOADDR5BEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[SHORTNAMEAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[SHORTNAMEBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[SIAMED'XPRESSAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[SIAMED'XPRESSBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[SIGMA-ALDRICHCORPORATIONAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[SIGMA-ALDRICHCORPORATIONBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[SIGNALPHARMACEUTICALSLLCAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[SIGNALPHARMACEUTICALSLLCBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[SINOPHARMCHEMICALREAGENTCO.LTDAFTER] [varchar] (255) COLLATE Latin1_General_BIN
NULL,
[SINOPHARMCHEMICALREAGENTCO.LTDBEFORE] [varchar] (255) COLLATE Latin1_General_BIN
NULL,
[SIREANALYTICALSYSTEMSAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[SIREANALYTICALSYSTEMSBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[SODEFAULTTYPEAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[SODEFAULTTYPEBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[SOLDPOSTALCODEAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[SOLDPOSTALCODEBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[SOLDTOADDR1AFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[SOLDTOADDR1BEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[SOLDTOADDR2AFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[SOLDTOADDR2BEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[SOLDTOADDR3AFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
```

```
[SOLDTOADDR3BEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[SOLDTOADDR3LOCAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[SOLDTOADDR3LOCBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[SOLDTOADDR4AFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[SOLDTOADDR4BEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[SOLDTOADDR5AFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[SOLDTOADDR5BEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[SUNPHARMACEUTICALINDUSTRIESLTDAFTER] [varchar] (255) COLLATE Latin1_General_BIN
NULL,
[SUNPHARMACEUTICALINDUSTRIESLTDBEFORE] [varchar] (255) COLLATE Latin1_General_BIN
NULL,
[SUNYEARSCIENTIFICINCAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[SUNYEARSCIENTIFICINCBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[SUTROBIOPHARMAAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[SUTROBIOPHARMABEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[SYMMETRIXBIOTECHPVT,LTD.AFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[SYMMETRIXBIOTECHPVT,LTD.BEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[SYNGENEINTERNATIONALLTDAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[SYNGENEINTERNATIONALLTDBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[TAXEXEMPTNUMBERAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[TAXEXEMPTNUMBERBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[TAXSTATUSAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[TAXSTATUSBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[TELEPHONEAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[TELEPHONEBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[TEVABIOPHARMACEUTICALSUSAINC.AFTER] [varchar] (255) COLLATE Latin1_General_BIN
NULL,
[TEVABIOPHARMACEUTICALSUSAINC.BEFORE] [varchar] (255) COLLATE Latin1_General_BIN
NULL,
[THERMOFISHERSCIENTIFICAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[THERMOFISHERSCIENTIFICBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[THEUNIVERSITYOFMANCHESTERAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[THEUNIVERSITYOFMANCHESTERBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[TNOEARTH,ENVIRONMENTAL&LSTNOAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[TNOEARTH,ENVIRONMENTAL&LSTNOBEFORE] [varchar] (255) COLLATE Latin1_General_BIN
NULL,
[TRANSACTIONNATUREAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[TRANSACTIONNATUREBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[TRANSACTIONNATURECAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[TRANSACTIONNATURECBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[TROPHOGENAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[TROPHOGENBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[TUMUNCHENAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[TUMUNCHENBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[UCBCELLTECHAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[UCBCELLTECHBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[UKVATFLAGAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[UKVATFLAGBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[UNIVERSITYCOLLEGELONDONAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[UNIVERSITYCOLLEGELONDONBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[UNIVERSITYOFBRITISHCOLUMBIAAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[UNIVERSITYOFBRITISHCOLUMBIABEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[UNIVERSITYOFCOPENHAGENAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[UNIVERSITYOFCOPENHAGENBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[UNIVERSITYOFGDANSKAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[UNIVERSITYOFGDANSKBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[UNIVERSITYOFHYDERABADAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[UNIVERSITYOFHYDERABADBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[UNIVERSITYOFKENTUCKYAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
```

```
[UNIVERSITYOFKENTUCKYBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[UNIVERSITYOFZURICHAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[UNIVERSITYOFZURICHBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[VERAXBIOMEDICALAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[VERAXBIOMEDICALBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[VITACYTELLCAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[VITACYTELLCBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[VIVOSTATA/SAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[VIVOSTATA/SBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[VWRINTERNATIONALAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[VWRINTERNATIONALBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[VWRINTERNATIONALBVBAAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[VWRINTERNATIONALBVBABEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[WAKOCHEMICALSGMBHAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[WAKOCHEMICALSGMBHBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[WEB-LABLABORATORYPRODUCTSCORPAFTER] [varchar] (255) COLLATE Latin1_General_BIN
NULL,
[WEB-LABLABORATORYPRODUCTSCORPBEFORE] [varchar] (255) COLLATE Latin1_General_BIN
NULL,
[WOOILSCIENCESAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[WOOILSCIENCESBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[YALEUNIVERSITYBCMMAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[YALEUNIVERSITYBCMMBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[YICHANGCHANGJIANGPHARMACEUTICALLTDAFTER] [varchar] (255) COLLATE Latin1_General_BIN
NULL,
[YICHANGCHANGJIANGPHARMACEUTICALLTDBEFORE] [varchar] (255) COLLATE Latin1_General_-
BIN NULL,
[YORKSHIREPROCESSTECHNOLOGYAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[YORKSHIREPROCESSTECHNOLOGYBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[ZYMETECHEHFAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[ZYMETECHEHFBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[ZYMEWORKSAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[ZYMEWORKSBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[ZYMOGENETICSINCAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[ZYMOGENETICSINCBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[ZYNGENIAAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[ZYNGENIABEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[DELTAPRECISIONLTDAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[DELTAPRECISIONLTDBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[KRISHGENBIOSYSTEMSAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[KRISHGENBIOSYSTEMSBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[LFBBIOTECHNOLOGIESAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[LFBBIOTECHNOLOGIESBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[MIKEBANUAAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[MIKEBANUABEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[MILLIPORESIGMAAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[MILLIPORESIGMABEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[PHILOGENS.P.AAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[PHILOGENS.P.ABEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[PNUVAXINCORPORATEDAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[PNUVAXINCORPORATEDBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[TEVAPHARMACEUTICALINDUSTRIESLTDAFTER] [varchar] (255) COLLATE Latin1_General_BIN
NULL,
[TEVAPHARMACEUTICALINDUSTRIESLTDBEFORE] [varchar] (255) COLLATE Latin1_General_BIN
NULL,
[UNIVERSITÉLAVALAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[UNIVERSITÉLAVALBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[VWRINTERNATIONALGMBHAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[VWRINTERNATIONALGMBHBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
```

```
[WHPARTNERSHIPLIMITEDAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[WHPARTNERSHIPLIMITEDBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[YICHANGHECCHANGJIANGPHARMACEUTICALCO.LTDAFTER] [varchar] (255) COLLATE Latin1_-
General_BIN NULL,
[YICHANGHECCHANGJIANGPHARMACEUTICALCO.LTDBEFORE] [varchar] (255) COLLATE Latin1_-
General_BIN NULL,
[YICHANGHECCHANGJIANGPHARMACEUTICALCOLTDAFTER] [varchar] (255) COLLATE Latin1_-
General_BIN NULL,
[YICHANGHECCHANGJIANGPHARMACEUTICALCOLTDBEFORE] [varchar] (255) COLLATE Latin1_-
General_BIN NULL,
[CELLTRICKSBIOTECHPVTLTDAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[CELLTRICKSBIOTECHPVTLTDBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[COOKREGENTECAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[COOKREGENTECBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[EPYGENBIOTECHAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[EPYGENBIOTECHBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[GREENCROSSCORPORATIONAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[GREENCROSSCORPORATIONBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[INSTITUTOSUPERIORTECNICO(IST-ID)AFTER] [varchar] (255) COLLATE Latin1_General_BIN
NULL,
[INSTITUTOSUPERIORTECNICO(IST-ID)BEFORE] [varchar] (255) COLLATE Latin1_General_BIN
NULL,
[LEEHYOBIOSCIENCECO.,LTD.AFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[LEEHYOBIOSCIENCECO.,LTD.BEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[ZEPTEONAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[ZEPTEONBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[TESTAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[TESTBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[HOCHSCHULEMANNHEIMINSTITUTFÜRBIOCHEMIEAFTER] [varchar] (255) COLLATE Latin1_-
General_BIN NULL,
[HOCHSCHULEMANNHEIMINSTITUTFÜRBIOCHEMIEBEFORE] [varchar] (255) COLLATE Latin1_-
General_BIN NULL,
[INTERPHARMBIOTEKLTDAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[INTERPHARMBIOTEKLTDBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL
) ON [PRIMARY]
GO
EXEC sp_addextendedproperty N'MS_Description', N'Logs from ArCustomer change logs',
'SCHEMA', N'History', 'TABLE', N'ArchiveArCustomer27OCT2016112011030', NULL, NULL
GO
```

Uses

[History]

Author: Johnson, Chris

[History].[ArchiveAssetMaster27OCT2016112011047]

MS_Description

Logs from AssetManager change logs

Columns

Name	Data Type	Max Length (Bytes)	Allow Nulls
TransactionDescription	varchar(150)	150	True
DatabaseName	varchar(150)	150	False
SignatureDateTime	datetime2	8	False
Operator	varchar(20)	20	False
ItemKey	varchar(150)	150	False
ComputerName	varchar(150)	150	True
ProgramName	varchar(100)	100	False
ConditionName	varchar(15)	15	True
AlreadyEntered	bit	1	True
ASSETCOSTAFTER	varchar(255)	255	True
ASSETCOSTBEFORE	varchar(255)	255	True
ASSETGROUPCODEAFTER	varchar(255)	255	True
ASSETGROUPCODEBEFORE	varchar(255)	255	True
BOOKVALDEPAFTER	varchar(255)	255	True
BOOKVALDEPBEFORE	varchar(255)	255	True
BRANCHAFTER	varchar(255)	255	True
BRANCHBEFORE	varchar(255)	255	True
BVASSETCOSTAFTER	varchar(255)	255	True
BVASSETCOSTBEFORE	varchar(255)	255	True
BVCURRENTVALUEAFTER	varchar(255)	255	True
BVCURRENTVALUEBEFORE	varchar(255)	255	True
BVDEPNTHISPERIODAFTER	varchar(255)	255	True
BVDEPNTHISPERIODBEFORE	varchar(255)	255	True
BVDEPNTHISYRAFTER	varchar(255)	255	True
BVDEPNTHISYRBEFORE	varchar(255)	255	True
DEPNSTARTPERIODAFTER	varchar(255)	255	True
DEPNSTARTPERIODBEFORE	varchar(255)	255	True
DEPNSTARTYEARAFTER	varchar(255)	255	True
DEPNSTARTYEARBEFORE	varchar(255)	255	True
DESCRIPTIONAFTER	varchar(255)	255	True
DESCRIPTIONBEFORE	varchar(255)	255	True
FIRSTINSTALDATEAFTER	varchar(255)	255	True
FIRSTINSTALDATEBEFORE	varchar(255)	255	True
IDCODEAFTER	varchar(255)	255	True

Author: Johnson, Chris

IDCODEBEFORE	varchar(255)	255	True
LOCATIONAFTER	varchar(255)	255	True
LOCATIONBEFORE	varchar(255)	255	True
NUMBEROFPERIODSAFTER	varchar(255)	255	True
NUMBEROFPERIODSBEFORE	varchar(255)	255	True
PURCHASEDATEAFTER	varchar(255)	255	True
PURCHASEDATEBEFORE	varchar(255)	255	True
PURCHASEPERIODAFTER	varchar(255)	255	True
PURCHASEPERIODBEFORE	varchar(255)	255	True
PURCHASEYEARAFTER	varchar(255)	255	True
PURCHASEYEARBEFORE	varchar(255)	255	True
RESIDUALVALUEAFTER	varchar(255)	255	True
RESIDUALVALUEBEFORE	varchar(255)	255	True
SOLDDATEAFTER	varchar(255)	255	True
SOLDDATEBEFORE	varchar(255)	255	True
SOLDPERIODAFTER	varchar(255)	255	True
SOLDPERIODBEFORE	varchar(255)	255	True
SOLDYEARAFTER	varchar(255)	255	True
SOLDYEARBEFORE	varchar(255)	255	True
STARTDEPDATEAFTER	varchar(255)	255	True
STARTDEPDATEBEFORE	varchar(255)	255	True
STATGLAFTER	varchar(255)	255	True
STATGLBEFORE	varchar(255)	255	True
SUPPLIERAFTER	varchar(255)	255	True
SUPPLIERBEFORE	varchar(255)	255	True
SUSPENDAFTER	varchar(255)	255	True
SUSPENDBEFORE	varchar(255)	255	True
TOTALPERIODSTDAFTER	varchar(255)	255	True
TOTALPERIODSTDBEFORE	varchar(255)	255	True
USERDEF3AFTER	varchar(255)	255	True
USERDEF3BEFORE	varchar(255)	255	True
VARYINGANNIVERSARYAFTER	varchar(255)	255	True
VARYINGANNIVERSARYBEFORE	varchar(255)	255	True
USERDEF4AFTER	varchar(255)	255	True
USERDEF4BEFORE	varchar(255)	255	True

SQL Script

```
CREATE TABLE [History].[ArchiveAssetMaster27OCT2016112011047]
(
[TransactionDescription] [varchar] (150) COLLATE Latin1_General_BIN NULL,
[DatabaseName] [varchar] (150) COLLATE Latin1_General_BIN NOT NULL,
[SignatureDateTime] [datetime2] NOT NULL,
[Operator] [varchar] (20) COLLATE Latin1_General_BIN NOT NULL,
[ItemKey] [varchar] (150) COLLATE Latin1_General_BIN NOT NULL,
```

Author: Johnson, Chris

```
[ComputerName] [varchar] (150) COLLATE Latin1_General_BIN NULL,
[ProgramName] [varchar] (100) COLLATE Latin1_General_BIN NOT NULL,
[ConditionName] [varchar] (15) COLLATE Latin1_General_BIN NULL,
[AlreadyEntered] [bit] NULL,
[ASSETCOSTAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[ASSETCOSTBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[ASSETGROUPCODEAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[ASSETGROUPCODEBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[BOOKVALDEPAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[BOOKVALDEPBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[BRANCHAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[BRANCHBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[BVASSETCOSTAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[BVASSETCOSTBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[BVCURRENTVALUEAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[BVCURRENTVALUEBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[BVDEPNTHISPERIODAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[BVDEPNTHISPERIODBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[BVDEPNTHISYRAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[BVDEPNTHISYRBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[DEPNSTARTPERIODAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[DEPNSTARTPERIODBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[DEPNSTARTYEARAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[DEPNSTARTYEARBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[DESCRIPTIONAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[DESCRIPTIONBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[FIRSTINSTALDATEAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[FIRSTINSTALDATEBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[IDCODEAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[IDCODEBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[LOCATIONAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[LOCATIONBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[NUMBEROFPERIODSAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[NUMBEROFPERIODSBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[PURCHASEDATEAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[PURCHASEDATEBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[PURCHASEPERIODAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[PURCHASEPERIODBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[PURCHASEYEARAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[PURCHASEYEARBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[RESIDUALVALUEAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[RESIDUALVALUEBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[SOLDDATEAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[SOLDDATEBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[SOLDPERIODAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[SOLDPERIODBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[SOLDYEARAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[SOLDYEARBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[STARTDEPDATEAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[STARTDEPDATEBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[STATGLAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[STATGLBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[SUPPLIERAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[SUPPLIERBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[SUSPENDAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[SUSPENDBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[TOTALPERIODSTDAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
```

Author: Johnson, Chris

```
[TOTALPERIODSTDBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[USERDEF3AFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[USERDEF3BEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[VARYINGANNIVERSARYAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[VARYINGANNIVERSARYBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[USERDEF4AFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[USERDEF4BEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL
) ON [PRIMARY]
GO
EXEC sp_addextendedproperty N'MS_Description', N'Logs from AssetManager change
logs', 'SCHEMA', N'History', 'TABLE', N'ArchiveAssetMaster27OCT2016112011047', NULL,
NULL
GO
```

Uses

[History]

Author: Johnson, Chris

[History].[ArchiveBomStructure27OCT2016112011093]

MS_Description

Logs from BOMStructure change logs

Columns

Name	Data Type	Max Length (Bytes)	Allow Nulls
TransactionDescription	varchar(150)	150	True
DatabaseName	varchar(150)	150	False
SignatureDateTime	datetime2	8	False
Operator	varchar(20)	20	False
ItemKey	varchar(150)	150	False
ComputerName	varchar(150)	150	True
ProgramName	varchar(100)	100	False
ConditionName	varchar(15)	15	True
AlreadyEntered	bit	1	True
NEWASSEMBLYPLACE	varchar(255)	255	True
NEWAUTONARRCODE	float	8	True
NEWCOMPONENTTYPE	varchar(255)	255	True
NEWCOMRELEASE	varchar(255)	255	True
NEWCOMVERSION	varchar(255)	255	True
NEWCREATESUBJOB	varchar(255)	255	True
NEWECCCONSUMPTION	varchar(255)	255	True
NEWFIXEDQTYPER	float	8	True
NEWFIXEDQTYPERENT	float	8	True
NEWFIXEDQTYPERFLAG	varchar(255)	255	True
NEWIGNOREFLOORFLAG	varchar(255)	255	True
NEWINCLKITISSUES	varchar(255)	255	True
NEWINCLUDEBATCH	varchar(255)	255	True
NEWINCLUDEFROMJOB	varchar(255)	255	True
NEWINCLUDETOJOB	varchar(255)	255	True
NEWINCSCRAPFLAG	varchar(255)	255	True
NEWITEMNUMBER	varchar(255)	255	True
NEWOPOFFSETFLAG	varchar(255)	255	True
NEWQTYPER	float	8	True
NEWQTYPERENT	float	8	True
NEWREASONFORCHG	varchar(255)	255	True
NEWREFDESIGNATOR	varchar(255)	255	True
NEWROLLUPCOSTFLAG	varchar(255)	255	True
NEWSCRAPPCT	float	8	True
NEWSCRAPQTY	float	8	True

Author: Johnson, Chris

NEWSCRAPQTYENT	float	8	True
NEWSOOPTIONFLAG	varchar(255)	255	True
NEWSOPRINTFLAG	varchar(255)	255	True
NEWSTRUCTOFFDATE	date	3	True
NEWSTRUCTONDATE	date	3	True
NEWUOMFLAG	varchar(255)	255	True
NEWWAREHOUSE	varchar(255)	255	True
NEWWETWEIGHTPERC	float	8	True
OLDASSEMBLYPLACE	varchar(255)	255	True
OLDAUTONARRCODE	float	8	True
OLDCOMPONENTTYPE	varchar(255)	255	True
OLDCOMRELEASE	varchar(255)	255	True
OLDCOMVERSION	varchar(255)	255	True
OLDCREATESUBJOB	varchar(255)	255	True
OLDECCCONSUMPTION	varchar(255)	255	True
OLDFIXEDQTYPER	float	8	True
OLDFIXEDQTYPERENT	float	8	True
OLDFIXEDQTYPERFLAG	varchar(255)	255	True
OLDIGNOREFLOORFLAG	varchar(255)	255	True
OLDINCLKITISSUES	varchar(255)	255	True
OLDINCLUDEBATCH	varchar(255)	255	True
OLDINCLUDEFROMJOB	varchar(255)	255	True
OLDINCLUDETOJOB	varchar(255)	255	True
OLDINCSCRAPFLAG	varchar(255)	255	True
OLDITEMNUMBER	varchar(255)	255	True
OLDOPOFFSETFLAG	varchar(255)	255	True
OLDQTYPER	float	8	True
OLDQTYPERENT	float	8	True
OLDREASONFORCHG	varchar(255)	255	True
OLDREFDESIGNATOR	varchar(255)	255	True
OLDROLLUPCOSTFLAG	varchar(255)	255	True
OLDSCRAPPCT	float	8	True
OLDSCRAPQTY	float	8	True
OLDSCRAPQTYENT	float	8	True
OLDSOOPTIONFLAG	varchar(255)	255	True
OLDSOPRINTFLAG	varchar(255)	255	True
OLDSTRUCTONDATE	date	3	True
OLDUOMFLAG	varchar(255)	255	True
OLDWAREHOUSE	varchar(255)	255	True
OLDWETWEIGHTPERC	float	8	True

SQL Script

```
CREATE TABLE [History].[ArchiveBomStructure27OCT2016112011093]
```

Author: Johnson, Chris

```
(
[TransactionDescription] [varchar] (150) COLLATE Latin1_General_BIN NULL,
[DatabaseName] [varchar] (150) COLLATE Latin1_General_BIN NOT NULL,
[SignatureDateTime] [datetime2] NOT NULL,
[Operator] [varchar] (20) COLLATE Latin1_General_BIN NOT NULL,
[ItemKey] [varchar] (150) COLLATE Latin1_General_BIN NOT NULL,
[ComputerName] [varchar] (150) COLLATE Latin1_General_BIN NULL,
[ProgramName] [varchar] (100) COLLATE Latin1_General_BIN NOT NULL,
[ConditionName] [varchar] (15) COLLATE Latin1_General_BIN NULL,
[AlreadyEntered] [bit] NULL,
[NEWASSEMBLYPLACE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[NEWAUTONARRCODE] [float] NULL,
[NEWCOMPONENTTYPE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[NEWCOMRELEASE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[NEWCOMVERSION] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[NEWCREATESUBJOB] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[NEWECCCONSUMPTION] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[NEWFIXEDQTYPER] [float] NULL,
[NEWFIXEDQTYPERENT] [float] NULL,
[NEWFIXEDQTYPERFLAG] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[NEWIGNOREFLOORFLAG] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[NEWINCLKITISSUES] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[NEWINCLUDEBATCH] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[NEWINCLUDEFROMJOB] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[NEWINCLUDETOJOB] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[NEWINCSCRAPFLAG] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[NEWITEMNUMBER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[NEWOPOFFSETFLAG] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[NEWQTYPER] [float] NULL,
[NEWQTYPERENT] [float] NULL,
[NEWREASONFORCHG] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[NEWREFDESIGNATOR] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[NEWROLLUPCOSTFLAG] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[NEWSCRAPPCT] [float] NULL,
[NEWSCRAPQTY] [float] NULL,
[NEWSCRAPQTYENT] [float] NULL,
[NEWSOOPTIONFLAG] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[NEWSOPRINTFLAG] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[NEWSTRUCTOFFDATE] [date] NULL,
[NEWSTRUCTONDATE] [date] NULL,
[NEWUOMFLAG] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[NEWWAREHOUSE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[NEWWETWEIGHTPERC] [float] NULL,
[OLDASSEMBLYPLACE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[OLDAUTONARRCODE] [float] NULL,
[OLDCOMPONENTTYPE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[OLDCOMRELEASE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[OLDCOMVERSION] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[OLDCREATESUBJOB] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[OLDECCCONSUMPTION] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[OLDFIXEDQTYPER] [float] NULL,
[OLDFIXEDQTYPERENT] [float] NULL,
[OLDFIXEDQTYPERFLAG] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[OLDIGNOREFLOORFLAG] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[OLDINCLKITISSUES] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[OLDINCLUDEBATCH] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[OLDINCLUDEFROMJOB] [varchar] (255) COLLATE Latin1_General_BIN NULL,
```

Author: Johnson, Chris

```
[OLDINCLUDETOJOB] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[OLDINCSCRAPFLAG] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[OLDITEMNUMBER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[OLDOPOFFSETFLAG] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[OLDQTYPER] [float] NULL,
[OLDQTYPERENT] [float] NULL,
[OLDREASONFORCHG] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[OLDREFDESIGNATOR] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[OLDROLLUPCOSTFLAG] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[OLDSCRAPPCT] [float] NULL,
[OLDSCRAPQTY] [float] NULL,
[OLDSCRAPQTYENT] [float] NULL,
[OLDSOOPTIONFLAG] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[OLDSOPRINTFLAG] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[OLDSTRUCTONDATE] [date] NULL,
[OLDUOMFLAG] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[OLDWAREHOUSE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[OLDWETWEIGHTPERC] [float] NULL
) ON [PRIMARY]
GO
EXEC sp_addextendedproperty N'MS_Description', N'Logs from BOMStructure change
logs', 'SCHEMA', N'History', 'TABLE', N'ArchiveBomStructure27OCT2016112011093',
NULL, NULL
GO
```

Uses

[History]

Author: Johnson, Chris

[History].[ArchiveExecForEachDBErrorLogs27OCT2016112011107]

MS_Description

place to capture details of errors generated by sp ExecForEachDB

Columns

Name	Data Type	Max Length (Bytes)	Allow Nulls	Identity	Default
LogID	bigint	8	False	1 - 1	
LogTime	datetime2	8	True		(getdate())
Error	nvarchar(2000)	4000	True		

SQL Script

```
CREATE TABLE [History].[ArchiveExecForEachDBErrorLogs27OCT2016112011107]
(
[LogID] [bigint] NOT NULL IDENTITY(1, 1),
[LogTime] [datetime2] NULL CONSTRAINT [DF__ExecForEa__LogTi__2FFA0313] DEFAULT
(getdate()),
[Error] [nvarchar] (2000) COLLATE Latin1_General_BIN NULL
) ON [PRIMARY]
GO
EXEC sp_addextendedproperty N'MS_Description', N'place to capture details of errors
generated by sp ExecForEachDB', 'SCHEMA', N'History', 'TABLE', N'ArchiveExecForEach-
DBErrorLogs27OCT2016112011107', NULL, NULL
GO
```

Uses

[History]

Author: Johnson, Chris

[History].[ArchiveExecForEachDBLogs27OCT2016112011103]

MS_Description

place to capture details of runs generated by sp ExecForEachDB

Columns

Name	Data Type	Max Length (Bytes)	Allow Nulls	Identity	Default
LogID	bigint	8	False	1 - 1	
LogTime	datetime2	8	True		(getdate())
Cmd	varchar(max)	max	True		

SQL Script

```
CREATE TABLE [History].[ArchiveExecForEachDBLogs27OCT2016112011103]
(
[LogID] [bigint] NOT NULL IDENTITY(1, 1),
[LogTime] [datetime2] NULL CONSTRAINT [DF__ExecForEa__LogTi__2E11BAA1] DEFAULT
(getdate()),
[Cmd] [varchar] (max) COLLATE Latin1_General_BIN NULL
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
GO
EXEC sp_addextendedproperty N'MS_Description', N'place to capture details of runs
generated by sp ExecForEachDB', 'SCHEMA', N'History', 'TABLE', N'ArchiveExecForEach-
DBLogs27OCT2016112011103', NULL, NULL
GO
```

Uses

[History]

[History].[ArchiveInvMaster27OCT2016112010997]

MS_Description

Logs from InvMaster change logs

Columns

Name	Data Type	Max Length (Bytes)	Allow Nulls
TransactionDescription	varchar(150)	150	True
DatabaseName	varchar(150)	150	False
SignatureDateTime	datetime2	8	False
Operator	varchar(20)	20	False
ItemKey	varchar(150)	150	False
ComputerName	varchar(150)	150	True
ProgramName	varchar(100)	100	False
ConditionName	varchar(15)	15	True
AlreadyEntered	bit	1	True
ABCANALYSISREQAFTER	varchar(255)	255	True
ABCANALYSISREQBEFORE	varchar(255)	255	True
ABCCOSTINGREQAFTER	varchar(255)	255	True
ABCCOSTINGREQBEFORE	varchar(255)	255	True
ALTERNATEKEY1AFTER	varchar(255)	255	True
ALTERNATEKEY1BEFORE	varchar(255)	255	True
ALTERNATEKEY2AFTER	varchar(255)	255	True
ALTERNATEKEY2BEFORE	varchar(255)	255	True
ALTERNATEUOMAFTER	varchar(255)	255	True
ALTERNATEUOMBEFORE	varchar(255)	255	True
ALTMETHODFLAGAFTER	varchar(255)	255	True
ALTMETHODFLAGBEFORE	varchar(255)	255	True
ALTREDUCTIONFLAGAFTER	varchar(255)	255	True
ALTREDUCTIONFLAGBEFORE	varchar(255)	255	True
ALTSISOFLAGAFTER	varchar(255)	255	True
ALTSISOFLAGBEFORE	varchar(255)	255	True
BASISAFTER	varchar(255)	255	True
BASISBEFORE	varchar(255)	255	True
BATCHBILLAFTER	varchar(255)	255	True
BATCHBILLBEFORE	varchar(255)	255	True
BULKISSUEFLAGAFTER	varchar(255)	255	True
BULKISSUEFLAGBEFORE	varchar(255)	255	True
BUYERAFTER	varchar(255)	255	True
BUYERBEFORE	varchar(255)	255	True
BUYINGRULEAFTER	varchar(255)	255	True

Author: Johnson, Chris

BUYINGRULEBEFORE	varchar(255)	255	True
CLEARINGFLAGAFTER	varchar(255)	255	True
CLEARINGFLAGBEFORE	varchar(255)	255	True
CONVFACTALTUOMAFTER	varchar(255)	255	True
CONVFACTALTUOMBEFORE	varchar(255)	255	True
CONVFACTOTHUOMAFTER	varchar(255)	255	True
CONVFACTOTHUOMBEFORE	varchar(255)	255	True
CONVMULDIVAFTER	varchar(255)	255	True
CONVMULDIVBEFORE	varchar(255)	255	True
COSTUOMAFTER	varchar(255)	255	True
COSTUOMBEFORE	varchar(255)	255	True
COUNTRYOFORIGINAFTER	varchar(255)	255	True
COUNTRYOFORIGINBEFORE	varchar(255)	255	True
CYCLECOUNTAFTER	varchar(255)	255	True
CYCLECOUNTBEFORE	varchar(255)	255	True
DECIMALSAFTER	varchar(255)	255	True
DECIMALSBEFORE	varchar(255)	255	True
DEMANDTIMEFENCEAFTER	varchar(255)	255	True
DEMANDTIMEFENCEBEFORE	varchar(255)	255	True
DESCRIPTIONAFTER	varchar(255)	255	True
DESCRIPTIONBEFORE	varchar(255)	255	True
DISTWAREHOUSETOUSEAFTER	varchar(255)	255	True
DISTWAREHOUSETOUSEBEFORE	varchar(255)	255	True
DOCKTOSTOCKAFTER	varchar(255)	255	True
DOCKTOSTOCKBEFORE	varchar(255)	255	True
DRAWOFFICENUMAFTER	varchar(255)	255	True
DRAWOFFICENUMBEFORE	varchar(255)	255	True
EBQAFTER	varchar(255)	255	True
EBQBEFORE	varchar(255)	255	True
EBQPANAFTER	varchar(255)	255	True
EBQPANBEFORE	varchar(255)	255	True
ECCFLAGAFTER	varchar(255)	255	True
ECCFLAGBEFORE	varchar(255)	255	True
ECCUSERAFTER	varchar(255)	255	True
ECCUSERBEFORE	varchar(255)	255	True
FIXOVERHEADAFTER	varchar(255)	255	True
FIXOVERHEADBEFORE	varchar(255)	255	True
FIXTIMEPERIODAFTER	varchar(255)	255	True
FIXTIMEPERIODBEFORE	varchar(255)	255	True
GROSSREQRULEAFTER	varchar(255)	255	True
GROSSREQRULEBEFORE	varchar(255)	255	True
GSTTAXCODEAFTER	varchar(255)	255	True
GSTTAXCODEBEFORE	varchar(255)	255	True
INCLINSTRVALIDAFTER	varchar(255)	255	True

INCLINSTRVALIDBEFORE	varchar(255)	255	True
INSPECTIONFLAGAFTER	varchar(255)	255	True
INSPECTIONFLAGBEFORE	varchar(255)	255	True
ISSMULTLOTSFLAGAFTER	varchar(255)	255	True
ISSMULTLOTSFLAGBEFORE	varchar(255)	255	True
JOBCLASSIFICATIONAFTER	varchar(255)	255	True
JOBCLASSIFICATIONBEFORE	varchar(255)	255	True
KITTYPEAFTER	varchar(255)	255	True
KITTYPEBEFORE	varchar(255)	255	True
LABOURCOSTAFTER	varchar(255)	255	True
LABOURCOSTBEFORE	varchar(255)	255	True
LCTREQUIREDAFTER	varchar(255)	255	True
LCTREQUIREDBEFORE	varchar(255)	255	True
LEADTIMEAFTER	varchar(255)	255	True
LEADTIMEBEFORE	varchar(255)	255	True
LISTPRICEAFTER	varchar(255)	255	True
LISTPRICEBEFORE	varchar(255)	255	True
LONGDESCAFTER	varchar(255)	255	True
LONGDESCBEFORE	varchar(255)	255	True
MAKETOORDERFLAGAFTER	varchar(255)	255	True
MAKETOORDERFLAGBEFORE	varchar(255)	255	True
MANUALCOSTFLAGAFTER	varchar(255)	255	True
MANUALCOSTFLAGBEFORE	varchar(255)	255	True
MANUFACTUREUOMAFTER	varchar(255)	255	True
MANUFACTUREUOMBEFORE	varchar(255)	255	True
MANUFLEADTIMEAFTER	varchar(255)	255	True
MANUFLEADTIMEBEFORE	varchar(255)	255	True
MASSAFTER	varchar(255)	255	True
MASSBEFORE	varchar(255)	255	True
MATERIALCOSTAFTER	varchar(255)	255	True
MATERIALCOSTBEFORE	varchar(255)	255	True
MINPRICEPCTAFTER	varchar(255)	255	True
MINPRICEPCTBEFORE	varchar(255)	255	True
MPSFLAGAFTER	varchar(255)	255	True
MPSFLAGBEFORE	varchar(255)	255	True
MULDIVAFTER	varchar(255)	255	True
MULDIVBEFORE	varchar(255)	255	True
MUMMULDIVAFTER	varchar(255)	255	True
MUMMULDIVBEFORE	varchar(255)	255	True
NOALTUNITAFTER	varchar(255)	255	True
NOALTUNITBEFORE	varchar(255)	255	True
ONHOLDREASONAFTER	varchar(255)	255	True
ONHOLDREASONBEFORE	varchar(255)	255	True
OTHERTAXCODEAFTER	varchar(255)	255	True

OTHERTAXCODEBEFORE	varchar(255)	255	True
OTHERUOMAFTER	varchar(255)	255	True
OTHERUOMBEFORE	varchar(255)	255	True
OUTPUTMASSFLAGAFTER	varchar(255)	255	True
OUTPUTMASSFLAGBEFORE	varchar(255)	255	True
PANSIZEAFTER	varchar(255)	255	True
PANSIZEBEFORE	varchar(255)	255	True
PARTCATEGORYAFTER	varchar(255)	255	True
PARTCATEGORYBEFORE	varchar(255)	255	True
PERCENTAGEYIELDAFTER	varchar(255)	255	True
PERCENTAGEYIELDBEFORE	varchar(255)	255	True
PHANTOMIFCOMPAFTER	varchar(255)	255	True
PHANTOMIFCOMPBEFORE	varchar(255)	255	True
PLANNERAFTER	varchar(255)	255	True
PLANNERBEFORE	varchar(255)	255	True
PRCINCLGSTAFTER	varchar(255)	255	True
PRCINCLGSTBEFORE	varchar(255)	255	True
PRICECATEGORYAFTER	varchar(255)	255	True
PRICECATEGORYBEFORE	varchar(255)	255	True
PRICEMETHODAFTER	varchar(255)	255	True
PRICEMETHODBEFORE	varchar(255)	255	True
PRICETYPEAFTER	varchar(255)	255	True
PRICETYPEBEFORE	varchar(255)	255	True
PRODUCTCLASSAFTER	varchar(255)	255	True
PRODUCTCLASSBEFORE	varchar(255)	255	True
PRODUCTGROUPAFTER	varchar(255)	255	True
PRODUCTGROUPBEFORE	varchar(255)	255	True
RELEASEAFTER	varchar(255)	255	True
RELEASEBEFORE	varchar(255)	255	True
RESOURCECODEAFTER	varchar(255)	255	True
RESOURCECODEBEFORE	varchar(255)	255	True
RETURNABLEITEMAFTER	varchar(255)	255	True
RETURNABLEITEMBEFORE	varchar(255)	255	True
SERENTRYATSALEAFTER	varchar(255)	255	True
SERENTRYATSALEBEFORE	varchar(255)	255	True
SERIALMETHODAFTER	varchar(255)	255	True
SERIALMETHODBEFORE	varchar(255)	255	True
SHELFLIFEAFTER	varchar(255)	255	True
SHELFLIFEBEFORE	varchar(255)	255	True
SPECIFICGRAVITYAFTER	varchar(255)	255	True
SPECIFICGRAVITYBEFORE	varchar(255)	255	True
STOCKANDALTUMAFTER	varchar(255)	255	True
STOCKANDALTUMBEFORE	varchar(255)	255	True
STOCKCODEAFTER	varchar(255)	255	True

STOCKCODEBEFORE	varchar(255)	255	True
STOCKMOVEMENTREQAFTER	varchar(255)	255	True
STOCKMOVEMENTREQBEFORE	varchar(255)	255	True
STOCKONHOLDAFTER	varchar(255)	255	True
STOCKONHOLDBEFORE	varchar(255)	255	True
STOCKUOMAFTER	varchar(255)	255	True
STOCKUOMBEFORE	varchar(255)	255	True
SUBCONTRACTCOSTAFTER	varchar(255)	255	True
SUBCONTRACTCOSTBEFORE	varchar(255)	255	True
SUPERCESSIONDATEAFTER	varchar(255)	255	True
SUPERCESSIONDATEBEFORE	varchar(255)	255	True
SUPPLEMENTARYCODEAFTER	varchar(255)	255	True
SUPPLEMENTARYCODEBEFORE	varchar(255)	255	True
SUPPLEMENTARYUNITAFTER	varchar(255)	255	True
SUPPLEMENTARYUNITBEFORE	varchar(255)	255	True
SUPPLIERAFTER	varchar(255)	255	True
SUPPLIERBEFORE	varchar(255)	255	True
TARIFFCODEAFTER	varchar(255)	255	True
TARIFFCODEBEFORE	varchar(255)	255	True
TAXCODEAFTER	varchar(255)	255	True
TAXCODEBEFORE	varchar(255)	255	True
TRACEABLETYPEAFTER	varchar(255)	255	True
TRACEABLETYPEBEFORE	varchar(255)	255	True
UNITQTYAFTER	varchar(255)	255	True
UNITQTYBEFORE	varchar(255)	255	True
USERFIELD1AFTER	varchar(255)	255	True
USERFIELD1BEFORE	varchar(255)	255	True
USERFIELD2AFTER	varchar(255)	255	True
USERFIELD2BEFORE	varchar(255)	255	True
USERFIELD3AFTER	varchar(255)	255	True
USERFIELD3BEFORE	varchar(255)	255	True
USERFIELD4AFTER	varchar(255)	255	True
USERFIELD4BEFORE	varchar(255)	255	True
USERFIELD5AFTER	varchar(255)	255	True
USERFIELD5BEFORE	varchar(255)	255	True
VARIABLEOVERHEADAFTER	varchar(255)	255	True
VARIABLEOVERHEADBEFORE	varchar(255)	255	True
VERSIONAFTER	varchar(255)	255	True
VERSIONBEFORE	varchar(255)	255	True
VOLUMEAFTER	varchar(255)	255	True
VOLUMEBEFORE	varchar(255)	255	True
WAREHOUSETOUSEAFTER	varchar(255)	255	True
WAREHOUSETOUSEBEFORE	varchar(255)	255	True
WIPCTLGLCODEAFTER	varchar(255)	255	True

WIPCTLGLCODEBEFORE	varchar(255)	255	True
WITHHOLDINGTAXEXPENSETYPEAFTER	varchar(255)	255	True
WITHHOLDINGTAXEXPENSETYPEBEFORE	varchar(255)	255	True

SQL Script

```
CREATE TABLE [History].[ArchiveInvMaster27OCT2016112010997]
(
[TransactionDescription] [varchar] (150) COLLATE Latin1_General_BIN NULL,
[DatabaseName] [varchar] (150) COLLATE Latin1_General_BIN NOT NULL,
[SignatureDateTime] [datetime2] NOT NULL,
[Operator] [varchar] (20) COLLATE Latin1_General_BIN NOT NULL,
[ItemKey] [varchar] (150) COLLATE Latin1_General_BIN NOT NULL,
[ComputerName] [varchar] (150) COLLATE Latin1_General_BIN NULL,
[ProgramName] [varchar] (100) COLLATE Latin1_General_BIN NOT NULL,
[ConditionName] [varchar] (15) COLLATE Latin1_General_BIN NULL,
[AlreadyEntered] [bit] NULL,
[ABCANALYSISREQAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[ABCANALYSISREQBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[ABCCOSTINGREQAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[ABCCOSTINGREQBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[ALTERNATEKEY1AFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[ALTERNATEKEY1BEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[ALTERNATEKEY2AFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[ALTERNATEKEY2BEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[ALTERNATEUOMAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[ALTERNATEUOMBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[ALTMETHODFLAGAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[ALTMETHODFLAGBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[ALTREDUCTIONFLAGAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[ALTREDUCTIONFLAGBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[ALTSISOFLAGAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[ALTSISOFLAGBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[BASISAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[BASISBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[BATCHBILLAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[BATCHBILLBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[BULKISSUEFLAGAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[BULKISSUEFLAGBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[BUYERAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[BUYERBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[BUYINGRULEAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[BUYINGRULEBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[CLEARINGFLAGAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[CLEARINGFLAGBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[CONVFACTALTUOMAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[CONVFACTALTUOMBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[CONVFACTOTHUOMAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[CONVFACTOTHUOMBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[CONVMULDIVAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[CONVMULDIVBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[COSTUOMAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[COSTUOMBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[COUNTRYOFORIGINAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[COUNTRYOFORIGINBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
```

Author: Johnson, Chris

```
[CYCLECOUNTAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[CYCLECOUNTBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[DECIMALSAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[DECIMALSBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[DEMANDTIMEFENCEAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[DEMANDTIMEFENCEBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[DESCRIPTIONAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[DESCRIPTIONBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[DISTWAREHOUSETOUSEAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[DISTWAREHOUSETOUSEBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[DOCKTOSTOCKAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[DOCKTOSTOCKBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[DRAWOFFICENUMAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[DRAWOFFICENUMBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[EBQAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[EBQBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[EBQPANAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[EBQPANBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[ECCFLAGAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[ECCFLAGBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[ECCUSERAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[ECCUSERBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[FIXOVERHEADAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[FIXOVERHEADBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[FIXTIMEPERIODAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[FIXTIMEPERIODBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[GROSSREQRULEAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[GROSSREQRULEBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[GSTTAXCODEAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[GSTTAXCODEBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[INCLINSTRVALIDAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[INCLINSTRVALIDBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[INSPECTIONFLAGAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[INSPECTIONFLAGBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[ISSMULTLOTSFLAGAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[ISSMULTLOTSFLAGBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[JOBCLASSIFICATIONAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[JOBCLASSIFICATIONBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[KITTYPEAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[KITTYPEBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[LABOURCOSTAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[LABOURCOSTBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[LCTREQUIREDAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[LCTREQUIREDBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[LEADTIMEAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[LEADTIMEBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[LISTPRICEAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[LISTPRICEBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[LONGDESCAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[LONGDESCBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[MAKETOORDERFLAGAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[MAKETOORDERFLAGBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[MANUALCOSTFLAGAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[MANUALCOSTFLAGBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[MANUFACTUREUOMAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[MANUFACTUREUOMBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[MANUFLEADTIMEAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
```

Author: Johnson, Chris

```
[MANUFLEADTIMEBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[MASSAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[MASSBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[MATERIALCOSTAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[MATERIALCOSTBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[MINPRICEPCTAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[MINPRICEPCTBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[MPSFLAGAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[MPSFLAGBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[MULDIVAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[MULDIVBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[MUMMULDIVAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[MUMMULDIVBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[NOALTUNITAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[NOALTUNITBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[ONHOLDREASONAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[ONHOLDREASONBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[OTHERTAXCODEAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[OTHERTAXCODEBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[OTHERUOMAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[OTHERUOMBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[OUTPUTMASSFLAGAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[OUTPUTMASSFLAGBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[PANSIZEAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[PANSIZEBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[PARTCATEGORYAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[PARTCATEGORYBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[PERCENTAGEYIELDAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[PERCENTAGEYIELDBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[PHANTOMIFCOMPAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[PHANTOMIFCOMPBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[PLANNERAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[PLANNERBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[PRCINCLGSTAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[PRCINCLGSTBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[PRICECATEGORYAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[PRICECATEGORYBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[PRICEMETHODAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[PRICEMETHODBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[PRICETYPEAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[PRICETYPEBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[PRODUCTCLASSAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[PRODUCTCLASSBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[PRODUCTGROUPAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[PRODUCTGROUPBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[RELEASEAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[RELEASEBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[RESOURCECODEAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[RESOURCECODEBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[RETURNABLEITEMAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[RETURNABLEITEMBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[SERENTRYATSALEAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[SERENTRYATSALEBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[SERIALMETHODAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[SERIALMETHODBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[SHELFLIFEAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[SHELFLIFEBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
```

Author: Johnson, Chris

```
[SPECIFICGRAVITYAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[SPECIFICGRAVITYBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[STOCKANDALTUMAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[STOCKANDALTUMBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[STOCKCODEAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[STOCKCODEBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[STOCKMOVEMENTREQAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[STOCKMOVEMENTREQBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[STOCKONHOLDAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[STOCKONHOLDBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[STOCKUOMAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[STOCKUOMBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[SUBCONTRACTCOSTAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[SUBCONTRACTCOSTBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[SUPERCESSIONDATEAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[SUPERCESSIONDATEBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[SUPPLEMENTARYCODEAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[SUPPLEMENTARYCODEBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[SUPPLEMENTARYUNITAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[SUPPLEMENTARYUNITBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[SUPPLIERAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[SUPPLIERBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[TARIFFCODEAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[TARIFFCODEBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[TAXCODEAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[TAXCODEBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[TRACEABLETYPEAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[TRACEABLETYPEBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[UNITQTYAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[UNITQTYBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[USERFIELD1AFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[USERFIELD1BEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[USERFIELD2AFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[USERFIELD2BEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[USERFIELD3AFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[USERFIELD3BEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[USERFIELD4AFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[USERFIELD4BEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[USERFIELD5AFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[USERFIELD5BEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[VARIABLEOVERHEADAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[VARIABLEOVERHEADBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[VERSIONAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[VERSIONBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[VOLUMEAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[VOLUMEBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[WAREHOUSETOUSEAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[WAREHOUSETOUSEBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[WIPCTLGLCODEAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[WIPCTLGLCODEBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[WITHHOLDINGTAXEXPENSETYPEAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[WITHHOLDINGTAXEXPENSETYPEBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL
) ON [PRIMARY]
GO
EXEC sp_addextendedproperty N'MS_Description', N'Logs from InvMaster change logs',
'SCHEMA', N'History', 'TABLE', N'ArchiveInvMaster27OCT2016112010997', NULL, NULL
GO
```

Author: Johnson, Chris

Uses

[History]

Author: Johnson, Chris

[History].[ArchiveInvWarehouse27OCT2016112011133]

Columns

Name	Data Type	Max Length (Bytes)	Allow Nulls	Identity
IID	int	4	False	1 - 1
TransactionDescription	varchar(150)	150	True	
SignatureDatetime	datetime2	8	False	
Operator	varchar(20)	20	False	
ProgramName	varchar(20)	20	False	
Ranking	bigint	8	True	
ItemKey	varchar(150)	150	False	
DatabaseName	varchar(150)	150	False	
BINLOCATION	varchar(255)	255	True	
CONDITIONDESCRIPTION	varchar(255)	255	True	
CURRENTCOMPANYDATE	datetime	8	True	
CURRENTCOMPANYID	varchar(255)	255	True	
CURRENTCOMPANYNAME	varchar(255)	255	True	
CURRENTOPERATINGSYSTEMDATE	datetime	8	True	
CURRENTOPERATINGSYSTEMTIME	varchar(255)	255	True	
DESTINATIONWAREHOUSE	varchar(255)	255	True	
INVWAREHOUSEKEY	varchar(255)	255	True	
JOURNALNUMBER	float	8	True	
LANGUAGECODE	varchar(255)	255	True	
LOTNUMBER	varchar(255)	255	True	
NOTATION	varchar(255)	255	True	
OPERATORCODE	varchar(255)	255	True	
OPERATORCURRENTGROUPCODE	varchar(255)	255	True	
OPERATOREMAILADDRESS	varchar(255)	255	True	
OPERATORLOCATION	varchar(255)	255	True	
OPERATORNAME	varchar(255)	255	True	
OPERATORPRIMARYGROUPCODE	varchar(255)	255	True	
POSTINGMONTH	float	8	True	
POSTINGYEAR	float	8	True	
PREVIOUSCOST	float	8	True	
PRODUCTCLASS	varchar(255)	255	True	
QUANTITY	float	8	True	
REFERENCE	varchar(255)	255	True	
SOURCEAPPLICATION	varchar(255)	255	True	
STOCKCODE	varchar(255)	255	True	
UNITCOST	float	8	True	
WAREHOUSE	varchar(255)	255	True	

Author: Johnson, Chris

SQL Script

```
CREATE TABLE [History].[ArchiveInvWarehouse27OCT2016112011133]
(
[IID] [int] NOT NULL IDENTITY(1, 1),
[TransactionDescription] [varchar] (150) COLLATE Latin1_General_BIN NULL,
[SignatureDatetime] [datetime2] NOT NULL,
[Operator] [varchar] (20) COLLATE Latin1_General_BIN NOT NULL,
[ProgramName] [varchar] (20) COLLATE Latin1_General_BIN NOT NULL,
[Ranking] [bigint] NULL,
[ItemKey] [varchar] (150) COLLATE Latin1_General_BIN NOT NULL,
[DatabaseName] [varchar] (150) COLLATE Latin1_General_BIN NOT NULL,
[BINLOCATION] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[CONDITIONDESCRIPTION] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[CURRENTCOMPANYDATE] [datetime] NULL,
[CURRENTCOMPANYID] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[CURRENTCOMPANYNAME] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[CURRENTOPERATINGSYSTEMDATE] [datetime] NULL,
[CURRENTOPERATINGSYSTEMTIME] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[DESTINATIONWAREHOUSE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[INVWAREHOUSEKEY] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[JOURNALNUMBER] [float] NULL,
[LANGUAGECODE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[LOTNUMBER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[NOTATION] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[OPERATORCODE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[OPERATORCURRENTGROUPCODE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[OPERATOREMAILADDRESS] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[OPERATORLOCATION] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[OPERATORNAME] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[OPERATORPRIMARYGROUPCODE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[POSTINGMONTH] [float] NULL,
[POSTINGYEAR] [float] NULL,
[PREVIOUSCOST] [float] NULL,
[PRODUCTCLASS] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[QUANTITY] [float] NULL,
[REFERENCE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[SOURCEAPPLICATION] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[STOCKCODE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[UNITCOST] [float] NULL,
[WAREHOUSE] [varchar] (255) COLLATE Latin1_General_BIN NULL
) ON [PRIMARY]
GO
```

Uses

[History]

Author: Johnson, Chris

[History].[ArchivePorMasterDetail27OCT2016112011023]

Columns

Name	Data Type	Max Length (Bytes)	Allow Nulls	Identity
PID	int	4	False	1 - 1
TransactionDescription	varchar(150)	150	True	
SignatureDatetime	datetime2	8	False	
Operator	varchar(20)	20	False	
ProgramName	varchar(20)	20	False	
Ranking	bigint	8	True	
ItemKey	varchar(150)	150	False	
DatabaseName	varchar(150)	150	False	
1STDISCOUNTPERCENT	float	8	True	
ALLOCATIONLINE	varchar(255)	255	True	
BINLOCATION	varchar(255)	255	True	
BUYER	varchar(255)	255	True	
CONDITIONDESCRIPTION	varchar(255)	255	True	
CONTRACTNUMBER	varchar(255)	255	True	
COSTBASIS	varchar(255)	255	True	
COSTINGMETHOD	varchar(255)	255	True	
COSTMULTIPLIER	float	8	True	
CREATEDFROM	varchar(255)	255	True	
CURRENTCOMPANYDATE	datetime	8	True	
CURRENTCOMPANYID	varchar(255)	255	True	
CURRENTCOMPANYNAME	varchar(255)	255	True	
CURRENTOPERATINGSYSTEMDATE	datetime	8	True	
CURRENTOPERATINGSYSTEMTIME	varchar(255)	255	True	
DESCRIPTION	varchar(255)	255	True	
FOREIGNPRICE	float	8	True	
GOODSRECEIVEDNUMBER	varchar(255)	255	True	
IMPORTFUNCTIONREQUESTED	varchar(255)	255	True	
ISSUETOJOB	float	8	True	
JOB	varchar(255)	255	True	
JOBALLOCATIONLINE	float	8	True	
JOBALLOCATIONSCOMPLETE	varchar(255)	255	True	
JOBCOMPLETE	varchar(255)	255	True	
JOBCONFIRMED	varchar(255)	255	True	
JOBONHOLD	varchar(255)	255	True	
JOURNAL	float	8	True	
JOURNALPOSTINGMONTH	float	8	True	
JOURNALPOSTINGYEAR	float	8	True	

Author: Johnson, Chris

LANGUAGECODE	varchar(255)	255	True	
LEDGERCODE	varchar(255)	255	True	
LINECOMPLETE	varchar(255)	255	True	
LINEDUEDATE	datetime	8	True	
LINEPREVIOUSLYCOMPLETE	varchar(255)	255	True	
LOT	varchar(255)	255	True	
LOTEXPIRYDATE	datetime	8	True	
LOTONFILE	varchar(255)	255	True	
MANYBINSUSEDTOPOST	varchar(255)	255	True	
NEWINVENTORYUNITCOST	float	8	True	
NONSTOCKEDFLAG	varchar(255)	255	True	
OPERATION	float	8	True	
OPERATORCODE	varchar(255)	255	True	
OPERATORCURRENTGROUPCODE	varchar(255)	255	True	
OPERATOREMAILADDRESS	varchar(255)	255	True	
OPERATORLOCATION	varchar(255)	255	True	
OPERATORNAME	varchar(255)	255	True	
OPERATORPRIMARYGROUPCODE	varchar(255)	255	True	
ORDERUNITOFMEASURE	varchar(255)	255	True	
ORIGINATOR	varchar(255)	255	True	
OUTSTANDINGORDERQUANTITY	float	8	True	
PORMASTERDETAILKEY	varchar(255)	255	True	
PREVIOUSINVENTORYUNITCOST	float	8	True	
PREVIOUSLEDGERCODE	varchar(255)	255	True	
PREVIOUSLINEDUEDATE	datetime	8	True	
PREVIOUSOUTSTANDINGORDERQUANTITY	float	8	True	
PREVIOUSPRICE	float	8	True	
PREVIOUSQUANTITY	float	8	True	
PRICE	float	8	True	
PRICEORDERRECEIVEDAT	float	8	True	
PRICINGUNITOFMEASURE	varchar(255)	255	True	
PRODUCTCLASS	varchar(255)	255	True	
PURCHASEORDER	varchar(255)	255	True	
PURCHASEORDERLINE	float	8	True	
PURCHASEORDERPRICE	float	8	True	
QUANTITY	float	8	True	
QUANTITYBEINGRECEIVED	float	8	True	
REQUISITION	varchar(255)	255	True	
REQUISITIONLINE	float	8	True	
REQUISITIONUSER	varchar(255)	255	True	
SALESORDER	varchar(255)	255	True	
SALESORDERCOMPLETE	varchar(255)	255	True	
SALESORDERLINE	float	8	True	
SOURCEAPPLICATION	varchar(255)	255	True	

STOCKCODE	varchar(255)	255	True	
STOCKDESCRIPTION	varchar(255)	255	True	
STOCKINGUNITOFMEASURE	varchar(255)	255	True	
SUPPLIER	varchar(255)	255	True	
SYSTEMPRICE	float	8	True	
WAREHOUSE	varchar(255)	255	True	
CAPEX	varchar(255)	255	True	
PREVIOUS1STDISCOUNTPERCENT	float	8	True	
PREVIOUSFOREIGNPRICE	float	8	True	
QUANTITYADVISED	float	8	True	
QUANTITYCOUNTED	float	8	True	
QUANTITYINSPECTED	float	8	True	
QUANTITYSTILLININSPECTION	float	8	True	

SQL Script

```
CREATE TABLE [History].[ArchivePorMasterDetail27OCT2016112011023]
(
[PID] [int] NOT NULL IDENTITY(1, 1),
[TransactionDescription] [varchar] (150) COLLATE Latin1_General_BIN NULL,
[SignatureDatetime] [datetime2] NOT NULL,
[Operator] [varchar] (20) COLLATE Latin1_General_BIN NOT NULL,
[ProgramName] [varchar] (20) COLLATE Latin1_General_BIN NOT NULL,
[Ranking] [bigint] NULL,
[ItemKey] [varchar] (150) COLLATE Latin1_General_BIN NOT NULL,
[DatabaseName] [varchar] (150) COLLATE Latin1_General_BIN NOT NULL,
[1STDISCOUNTPERCENT] [float] NULL,
[ALLOCATIONLINE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[BINLOCATION] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[BUYER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[CONDITIONDESCRIPTION] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[CONTRACTNUMBER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[COSTBASIS] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[COSTINGMETHOD] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[COSTMULTIPLIER] [float] NULL,
[CREATEDFROM] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[CURRENTCOMPANYDATE] [datetime] NULL,
[CURRENTCOMPANYID] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[CURRENTCOMPANYNAME] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[CURRENTOPERATINGSYSTEMDATE] [datetime] NULL,
[CURRENTOPERATINGSYSTEMTIME] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[DESCRIPTION] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[FOREIGNPRICE] [float] NULL,
[GOODSRECEIVEDNUMBER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[IMPORTFUNCTIONREQUESTED] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[ISSUETOJOB] [float] NULL,
[JOB] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[JOBALLOCATIONLINE] [float] NULL,
[JOBALLOCATIONSCOMPLETE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[JOBCOMPLETE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[JOBCONFIRMED] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[JOBONHOLD] [varchar] (255) COLLATE Latin1_General_BIN NULL,
```

Author: Johnson, Chris

```
[JOURNAL] [float] NULL,
[JOURNALPOSTINGMONTH] [float] NULL,
[JOURNALPOSTINGYEAR] [float] NULL,
[LANGUAGECODE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[LEDGERCODE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[LINECOMPLETE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[LINEDUEDATE] [datetime] NULL,
[LINEPREVIOUSLYCOMPLETE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[LOT] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[LOTEXPIRYDATE] [datetime] NULL,
[LOTONFILE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[MANYBINSUSEDTOPOST] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[NEWINVENTORYUNITCOST] [float] NULL,
[NONSTOCKEDFLAG] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[OPERATION] [float] NULL,
[OPERATORCODE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[OPERATORCURRENTGROUPCODE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[OPERATOREMAILADDRESS] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[OPERATORLOCATION] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[OPERATORNAME] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[OPERATORPRIMARYGROUPCODE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[ORDERUNITOFMEASURE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[ORIGINATOR] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[OUTSTANDINGORDERQUANTITY] [float] NULL,
[PORMASTERDETAILKEY] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[PREVIOUSINVENTORYUNITCOST] [float] NULL,
[PREVIOUSLEDGERCODE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[PREVIOUSLINEDUEDATE] [datetime] NULL,
[PREVIOUSOUTSTANDINGORDERQUANTITY] [float] NULL,
[PREVIOUSPRICE] [float] NULL,
[PREVIOUSQUANTITY] [float] NULL,
[PRICE] [float] NULL,
[PRICEORDERRECEIVEDAT] [float] NULL,
[PRICINGUNITOFMEASURE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[PRODUCTCLASS] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[PURCHASEORDER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[PURCHASEORDERLINE] [float] NULL,
[PURCHASEORDERPRICE] [float] NULL,
[QUANTITY] [float] NULL,
[QUANTITYBEINGRECEIVED] [float] NULL,
[REQUISITION] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[REQUISITIONLINE] [float] NULL,
[REQUISITIONUSER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[SALESORDER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[SALESORDERCOMPLETE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[SALESORDERLINE] [float] NULL,
[SOURCEAPPLICATION] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[STOCKCODE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[STOCKDESCRIPTION] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[STOCKINGUNITOFMEASURE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[SUPPLIER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[SYSTEMPRICE] [float] NULL,
[WAREHOUSE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[CAPEX] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[PREVIOUS1STDISCOUNTPERCENT] [float] NULL,
[PREVIOUSFOREIGNPRICE] [float] NULL,
[QUANTITYADVISED] [float] NULL,
```

Author: Johnson, Chris

```
[QUANTITYCOUNTED] [float] NULL,
[QUANTITYINSPECTED] [float] NULL,
[QUANTITYSTILLININSPECTION] [float] NULL
) ON [PRIMARY]
GO
```

Uses

[History]

[History].[ArchivePorMasterHdr27OCT2016112011037]

Columns

Name	Data Type	Max Length (Bytes)	Allow Nulls	Identity
PID	int	4	False	1 - 1
TransactionDescription	varchar(150)	150	True	
SignatureDatetime	datetime2	8	False	
Operator	varchar(20)	20	False	
ProgramName	varchar(20)	20	False	
Ranking	bigint	8	True	
ItemKey	varchar(150)	150	False	
DatabaseName	varchar(150)	150	False	
1STDISCOUNTPERCENT	float	8	True	
BUYER	varchar(255)	255	True	
CONDITIONDESCRIPTION	varchar(255)	255	True	
CURRENTCOMPANYDATE	datetime	8	True	
CURRENTCOMPANYID	varchar(255)	255	True	
CURRENTCOMPANYNAME	varchar(255)	255	True	
CURRENTOPERATINGSYSTEMDATE	datetime	8	True	
CURRENTOPERATINGSYSTEMTIME	varchar(255)	255	True	
CUSTOMER	varchar(255)	255	True	
LANGUAGECODE	varchar(255)	255	True	
MINIMUMORDERMASS	float	8	True	
MINIMUMORDERVALUE	float	8	True	
MINIMUMORDERVOLUME	float	8	True	
OPERATORCODE	varchar(255)	255	True	
OPERATORCURRENTGROUPCODE	varchar(255)	255	True	
OPERATOREMAILADDRESS	varchar(255)	255	True	
OPERATORLOCATION	varchar(255)	255	True	
OPERATORNAME	varchar(255)	255	True	
OPERATORPRIMARYGROUPCODE	varchar(255)	255	True	
ORDERDUEDATE	datetime	8	True	
ORDERMASS	float	8	True	
ORDERVALUE	float	8	True	
ORDERVOLUME	float	8	True	
PURCHASEORDER	varchar(255)	255	True	
REQUISITIONFLAG	varchar(255)	255	True	
SOURCEAPPLICATION	varchar(255)	255	True	
SUPPLIER	varchar(255)	255	True	

Author: Johnson, Chris

SQL Script

```
CREATE TABLE [History].[ArchivePorMasterHdr27OCT2016112011037]
(
[PID] [int] NOT NULL IDENTITY(1, 1),
[TransactionDescription] [varchar] (150) COLLATE Latin1_General_BIN NULL,
[SignatureDatetime] [datetime2] NOT NULL,
[Operator] [varchar] (20) COLLATE Latin1_General_BIN NOT NULL,
[ProgramName] [varchar] (20) COLLATE Latin1_General_BIN NOT NULL,
[Ranking] [bigint] NULL,
[ItemKey] [varchar] (150) COLLATE Latin1_General_BIN NOT NULL,
[DatabaseName] [varchar] (150) COLLATE Latin1_General_BIN NOT NULL,
[1STDISCOUNTPERCENT] [float] NULL,
[BUYER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[CONDITIONDESCRIPTION] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[CURRENTCOMPANYDATE] [datetime] NULL,
[CURRENTCOMPANYID] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[CURRENTCOMPANYNAME] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[CURRENTOPERATINGSYSTEMDATE] [datetime] NULL,
[CURRENTOPERATINGSYSTEMTIME] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[CUSTOMER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[LANGUAGECODE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[MINIMUMORDERMASS] [float] NULL,
[MINIMUMORDERVALUE] [float] NULL,
[MINIMUMORDERVOLUME] [float] NULL,
[OPERATORCODE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[OPERATORCURRENTGROUPCODE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[OPERATOREMAILADDRESS] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[OPERATORLOCATION] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[OPERATORNAME] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[OPERATORPRIMARYGROUPCODE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[ORDERDUEDATE] [datetime] NULL,
[ORDERMASS] [float] NULL,
[ORDERVALUE] [float] NULL,
[ORDERVOLUME] [float] NULL,
[PURCHASEORDER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[REQUISITIONFLAG] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[SOURCEAPPLICATION] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[SUPPLIER] [varchar] (255) COLLATE Latin1_General_BIN NULL
) ON [PRIMARY]
GO
```

Uses

[History]

Author: Johnson, Chris

[History].[ArchiveRedTagLogs27OCT2016112011100]

MS_Description

history of reports run

Columns

Key	Name	Data Type	Max Length (Bytes)	Allow Nulls	Identity	Default
	TagID	int	4	False	1 - 1	
	TagDatetime	datetime2	8	True		(getdate())
	StoredProcDb	varchar(255)	255	True		
	StoredProcSchema	varchar(255)	255	True		
(2)	StoredProcName	varchar(255)	255	True		
FK	UsedByType	char(1)	1	True		
(3)	UsedByName	varchar(500)	500	True		
	UsedByDb	varchar(255)	255	True		

Indexes

Name	Columns
IX_RedTagLogs_StoredProcName	UsedByName, StoredProcName
IX_RedTagLogs_StoredProcSchema	TagID, StoredProcName, UsedByName, StoredProcSchema
IX_RedTagLogs_UsedByName	TagDatetime, UsedByType, UsedByName

Foreign Keys

Name	Columns
FK__RedTagLog__UsedB__3A779186	UsedByType->[Lookups].[RedTagsUsedByType].[UsedByType]

SQL Script

```
CREATE TABLE [History].[ArchiveRedTagLogs27OCT2016112011100]
(
[TagID] [int] NOT NULL IDENTITY(1, 1),
[TagDatetime] [datetime2] NULL CONSTRAINT [DF__RedTagLog__TagDa__39836D4D] DEFAULT
(getdate()),
[StoredProcDb] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[StoredProcSchema] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[StoredProcName] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[UsedByType] [char] (1) COLLATE Latin1_General_BIN NULL,
[UsedByName] [varchar] (500) COLLATE Latin1_General_BIN NULL,
[UsedByDb] [varchar] (255) COLLATE Latin1_General_BIN NULL
) ON [PRIMARY]
```

Author: Johnson, Chris

```
GO
CREATE NONCLUSTERED INDEX [IX_RedTagLogs_StoredProcName] ON [History].[ArchiveRedTag-
Logs27OCT2016112011100] ([StoredProcName]) INCLUDE ([UsedByName]) ON [PRIMARY]
GO
CREATE NONCLUSTERED INDEX [IX_RedTagLogs_StoredProcSchema] ON [History].[ArchiveRed-
TagLogs27OCT2016112011100] ([StoredProcSchema]) INCLUDE ([StoredProcName], [TagID],
[UsedByName]) ON [PRIMARY]
GO
CREATE NONCLUSTERED INDEX [IX_RedTagLogs_UsedByName] ON [History].[ArchiveRedTag-
Logs27OCT2016112011100] ([UsedByName]) INCLUDE ([TagDatetime], [UsedByType]) ON
[PRIMARY]
GO
ALTER TABLE [History].[ArchiveRedTagLogs27OCT2016112011100] ADD CONSTRAINT [FK__Red-
TagLog__UsedB__3A779186] FOREIGN KEY ([UsedByType]) REFERENCES [Lookups].[RedTags-
UsedByType] ([UsedByType])
GO
EXEC sp_addextendedproperty N'MS_Description', N'history of reports run', 'SCHEMA',
N'History', 'TABLE', N'ArchiveRedTagLogs27OCT2016112011100', NULL, NULL
GO
```

Uses

[Lookups].[RedTagsUsedByType]
[History]

Author: Johnson, Chris

[History].[ArchiveReqDetail27OCT2016112011070]

Columns

Name	Data Type	Max Length (Bytes)	Allow Nulls	Identity
RID	int	4	False	1 - 1
TransactionDescription	varchar(150)	150	True	
SignatureDatetime	datetime2	8	False	
Operator	varchar(20)	20	False	
ProgramName	varchar(20)	20	False	
Ranking	bigint	8	True	
ItemKey	varchar(150)	150	False	
DatabaseName	varchar(150)	150	False	
ORIGINATOR	varchar(255)	255	True	
REQUISITION	varchar(255)	255	True	
REQUISITIONLINE	float	8	True	
REQUISITIONUSER	varchar(255)	255	True	
ROUTEDTOUSER	varchar(255)	255	True	
CONDITIONDESCRIPTION	varchar(255)	255	True	
CURRENTCOMPANYDATE	datetime	8	True	
CURRENTCOMPANYID	varchar(255)	255	True	
CURRENTCOMPANYNAME	varchar(255)	255	True	
CURRENTHOLDER	varchar(255)	255	True	
CURRENTOPERATINGSYSTEMDATE	datetime	8	True	
CURRENTOPERATINGSYSTEMTIME	varchar(255)	255	True	
DATEREQUISITIONRAISED	datetime	8	True	
LANGUAGECODE	varchar(255)	255	True	
LINEVALUE	float	8	True	
NEWASSETCAPEX	varchar(255)	255	True	
NEWCAPEXNUMBER	varchar(255)	255	True	
NEWDUEDATE	datetime	8	True	
NEWLEDGERCODE	varchar(255)	255	True	
NEWLINEVALUE	float	8	True	
NEWORDERQUANTITY	float	8	True	
NEWPRICE	float	8	True	
NEWPRODUCTCLASS	varchar(255)	255	True	
NEWSUPPLIER	varchar(255)	255	True	
OPERATORCODE	varchar(255)	255	True	
OPERATORCURRENTGROUPCODE	varchar(255)	255	True	
OPERATOREMAILADDRESS	varchar(255)	255	True	
OPERATORLOCATION	varchar(255)	255	True	
OPERATORNAME	varchar(255)	255	True	

Author: Johnson, Chris

OPERATORPRIMARYGROUPCODE	varchar(255)	255	True	
PREVIOUSCAPEXNUMBER	varchar(255)	255	True	
PREVIOUSDUEDATE	datetime	8	True	
PREVIOUSLEDGERCODE	varchar(255)	255	True	
PREVIOUSLINEVALUE	float	8	True	
PREVIOUSORDERQUANTITY	float	8	True	
PREVIOUSPRICE	float	8	True	
PREVIOUSPRODUCTCLASS	varchar(255)	255	True	
PREVIOUSSUPPLIER	varchar(255)	255	True	
REQUISITIONSTATUS	varchar(255)	255	True	
ROUTEDTOREQUISITIONUSER	varchar(255)	255	True	
SOURCEAPPLICATION	varchar(255)	255	True	
STOCKCODE	varchar(255)	255	True	
SUPPLIER	varchar(255)	255	True	
WAREHOUSE	varchar(255)	255	True	

SQL Script

```
CREATE TABLE [History].[ArchiveReqDetail27OCT2016112011070]
(
[RID] [int] NOT NULL IDENTITY(1, 1),
[TransactionDescription] [varchar] (150) COLLATE Latin1_General_BIN NULL,
[SignatureDatetime] [datetime2] NOT NULL,
[Operator] [varchar] (20) COLLATE Latin1_General_BIN NOT NULL,
[ProgramName] [varchar] (20) COLLATE Latin1_General_BIN NOT NULL,
[Ranking] [bigint] NULL,
[ItemKey] [varchar] (150) COLLATE Latin1_General_BIN NOT NULL,
[DatabaseName] [varchar] (150) COLLATE Latin1_General_BIN NOT NULL,
[ORIGINATOR] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[REQUISITION] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[REQUISITIONLINE] [float] NULL,
[REQUISITIONUSER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[ROUTEDTOUSER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[CONDITIONDESCRIPTION] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[CURRENTCOMPANYDATE] [datetime] NULL,
[CURRENTCOMPANYID] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[CURRENTCOMPANYNAME] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[CURRENTHOLDER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[CURRENTOPERATINGSYSTEMDATE] [datetime] NULL,
[CURRENTOPERATINGSYSTEMTIME] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[DATEREQUISITIONRAISED] [datetime] NULL,
[LANGUAGECODE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[LINEVALUE] [float] NULL,
[NEWASSETCAPEX] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[NEWCAPEXNUMBER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[NEWDUEDATE] [datetime] NULL,
[NEWLEDGERCODE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[NEWLINEVALUE] [float] NULL,
[NEWORDERQUANTITY] [float] NULL,
[NEWPRICE] [float] NULL,
[NEWPRODUCTCLASS] [varchar] (255) COLLATE Latin1_General_BIN NULL,
```

Author: Johnson, Chris

```
[NEWSUPPLIER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[OPERATORCODE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[OPERATORCURRENTGROUPCODE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[OPERATOREMAILADDRESS] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[OPERATORLOCATION] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[OPERATORNAME] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[OPERATORPRIMARYGROUPCODE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[PREVIOUSCAPEXNUMBER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[PREVIOUSDUEDATE] [datetime] NULL,
[PREVIOUSLEDGERCODE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[PREVIOUSLINEVALUE] [float] NULL,
[PREVIOUSORDERQUANTITY] [float] NULL,
[PREVIOUSPRICE] [float] NULL,
[PREVIOUSPRODUCTCLASS] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[PREVIOUSSUPPLIER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[REQUISITIONSTATUS] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[ROUTEDTOREQUISITIONUSER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[SOURCEAPPLICATION] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[STOCKCODE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[SUPPLIER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[WAREHOUSE] [varchar] (255) COLLATE Latin1_General_BIN NULL
) ON [PRIMARY]
GO
```

Uses

[History]

[History].[ArchiveReqHeader27OCT2016112011077]

Columns

Name	Data Type	Max Length (Bytes)	Allow Nulls	Identity
RID	int	4	False	1 - 1
TransactionDescription	varchar(150)	150	True	
SignatureDatetime	datetime2	8	False	
Operator	varchar(20)	20	False	
ProgramName	varchar(20)	20	False	
Ranking	bigint	8	True	
ItemKey	varchar(150)	150	False	
DatabaseName	varchar(150)	150	False	
CONDITIONDESCRIPTION	varchar(255)	255	True	
DATEREQUISITIONRAISED	datetime	8	True	
OPERATORCODE	varchar(255)	255	True	
OPERATORNAME	varchar(255)	255	True	
ORIGINATOR	varchar(255)	255	True	
REQUISITION	varchar(255)	255	True	
REQUISITIONUSER	varchar(255)	255	True	
ROUTEDTOREQUISITIONUSER	varchar(255)	255	True	
VALUEOFREQUISITION	float	8	True	

SQL Script

```
CREATE TABLE [History].[ArchiveReqHeader27OCT2016112011077]
(
[RID] [int] NOT NULL IDENTITY(1, 1),
[TransactionDescription] [varchar] (150) COLLATE Latin1_General_BIN NULL,
[SignatureDatetime] [datetime2] NOT NULL,
[Operator] [varchar] (20) COLLATE Latin1_General_BIN NOT NULL,
[ProgramName] [varchar] (20) COLLATE Latin1_General_BIN NOT NULL,
[Ranking] [bigint] NULL,
[ItemKey] [varchar] (150) COLLATE Latin1_General_BIN NOT NULL,
[DatabaseName] [varchar] (150) COLLATE Latin1_General_BIN NOT NULL,
[CONDITIONDESCRIPTION] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[DATEREQUISITIONRAISED] [datetime] NULL,
[OPERATORCODE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[OPERATORNAME] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[ORIGINATOR] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[REQUISITION] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[REQUISITIONUSER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[ROUTEDTOREQUISITIONUSER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[VALUEOFREQUISITION] [float] NULL
) ON [PRIMARY]
GO
```

Author: Johnson, Chris

Uses

[History]

Author: Johnson, Chris

[History].[ArchiveSorMaster27OCT2016112011007]

Columns

Name	Data Type	Max Length (Bytes)	Allow Nulls	Identity
SID	int	4	False	1 - 1
TransactionDescription	varchar(150)	150	True	
SignatureDatetime	datetime2	8	False	
Operator	varchar(20)	20	False	
ProgramName	varchar(20)	20	False	
Ranking	bigint	8	True	
ItemKey	varchar(150)	150	False	
DatabaseName	varchar(150)	150	False	
1STTRADEDISCOUNTPERCENTAGE	float	8	True	
2NDTRADEDISCOUNTPERCENTAGE	float	8	True	
3RDTRADEDISCOUNTPERCENTAGE	float	8	True	
ALTERNATEKEY	varchar(255)	255	True	
BRANCHCODE	varchar(255)	255	True	
CONDITIONDESCRIPTION	varchar(255)	255	True	
CURRENTCOMPANYDATE	datetime	8	True	
CURRENTCOMPANYID	varchar(255)	255	True	
CURRENTCOMPANYNAME	varchar(255)	255	True	
CURRENTOPERATINGSYSTEMDATE	datetime	8	True	
CURRENTOPERATINGSYSTEMTIME	varchar(255)	255	True	
CUSTOMERCODE	varchar(255)	255	True	
CUSTOMERP/ONUMBER	varchar(255)	255	True	
DELIVERYNOTENUMBER	varchar(255)	255	True	
DOCUMENTFORMAT	varchar(255)	255	True	
DOCUMENTTYPE	varchar(255)	255	True	
GEOGRAPHICAREA	varchar(255)	255	True	
GTRNUMBERFORSCT	varchar(255)	255	True	
INVOICEDATE	datetime	8	True	
INVOICEDATEDAYSDIFFERENT	float	8	True	
INVOICEDATEENTERED	datetime	8	True	
INVOICENUMBER	varchar(255)	255	True	
INVOICETERMSCODE	varchar(255)	255	True	
LANGUAGECODE	varchar(255)	255	True	
NEWORDERSTATUS	varchar(255)	255	True	
OPERATORCODE	varchar(255)	255	True	
OPERATORCURRENTGROUPCODE	varchar(255)	255	True	
OPERATOREMAILADDRESS	varchar(255)	255	True	
OPERATORLOCATION	varchar(255)	255	True	

Author: Johnson, Chris

OPERATORNAME	varchar(255)	255	True	
OPERATORPRIMARYGROUPCODE	varchar(255)	255	True	
ORDERDATE	datetime	8	True	
ORDERDISCOUNTVALUE	float	8	True	
ORDERSTATUS	varchar(255)	255	True	
ORDERTYPE	varchar(255)	255	True	
ORDERVALUE	float	8	True	
PRINTOREMAILFLAG	varchar(255)	255	True	
REPRINTFLAG	varchar(255)	255	True	
REQUESTEDSHIPDATE	datetime	8	True	
SALESORDER	varchar(255)	255	True	
SALESORDERNUMBER	varchar(255)	255	True	
SALESPERSON	varchar(255)	255	True	
SHIPPINGINSTRUCTIONS	varchar(255)	255	True	
SOURCEAPPLICATION	varchar(255)	255	True	
SPECIALINSTRUCTIONS	varchar(255)	255	True	
TYPEOFORDER	varchar(255)	255	True	

SQL Script

```
CREATE TABLE [History].[ArchiveSorMaster27OCT2016112011007]
(
[SID] [int] NOT NULL IDENTITY(1, 1),
[TransactionDescription] [varchar] (150) COLLATE Latin1_General_BIN NULL,
[SignatureDatetime] [datetime2] NOT NULL,
[Operator] [varchar] (20) COLLATE Latin1_General_BIN NOT NULL,
[ProgramName] [varchar] (20) COLLATE Latin1_General_BIN NOT NULL,
[Ranking] [bigint] NULL,
[ItemKey] [varchar] (150) COLLATE Latin1_General_BIN NOT NULL,
[DatabaseName] [varchar] (150) COLLATE Latin1_General_BIN NOT NULL,
[1STTRADEDISCOUNTPERCENTAGE] [float] NULL,
[2NDTRADEDISCOUNTPERCENTAGE] [float] NULL,
[3RDTRADEDISCOUNTPERCENTAGE] [float] NULL,
[ALTERNATEKEY] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[BRANCHCODE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[CONDITIONDESCRIPTION] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[CURRENTCOMPANYDATE] [datetime] NULL,
[CURRENTCOMPANYID] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[CURRENTCOMPANYNAME] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[CURRENTOPERATINGSYSTEMDATE] [datetime] NULL,
[CURRENTOPERATINGSYSTEMTIME] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[CUSTOMERCODE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[CUSTOMERP/ONUMBER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[DELIVERYNOTENUMBER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[DOCUMENTFORMAT] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[DOCUMENTTYPE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[GEOGRAPHICAREA] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[GTRNUMBERFORSCT] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[INVOICEDATE] [datetime] NULL,
[INVOICEDATEDAYSDIFFERENT] [float] NULL,
[INVOICEDATEENTERED] [datetime] NULL,
```

Author: Johnson, Chris

```
[INVOICENUMBER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[INVOICETERMSCODE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[LANGUAGECODE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[NEWORDERSTATUS] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[OPERATORCODE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[OPERATORCURRENTGROUPCODE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[OPERATOREMAILADDRESS] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[OPERATORLOCATION] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[OPERATORNAME] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[OPERATORPRIMARYGROUPCODE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[ORDERDATE] [datetime] NULL,
[ORDERDISCOUNTVALUE] [float] NULL,
[ORDERSTATUS] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[ORDERTYPE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[ORDERVALUE] [float] NULL,
[PRINTOREMAILFLAG] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[REPRINTFLAG] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[REQUESTEDSHIPDATE] [datetime] NULL,
[SALESORDER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[SALESORDERNUMBER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[SALESPERSON] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[SHIPPINGINSTRUCTIONS] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[SOURCEAPPLICATION] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[SPECIALINSTRUCTIONS] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[TYPEOFORDER] [varchar] (255) COLLATE Latin1_General_BIN NULL
) ON [PRIMARY]
GO
```

Uses

[History]

Author: Johnson, Chris

[History].[ArchiveTblCurrency27OCT2016112011017]

MS_Description

Logs from TblCurrency change logs

Columns

Name	Data Type	Max Length (Bytes)	Allow Nulls
TransactionDescription	varchar(150)	150	True
DatabaseName	varchar(150)	150	False
SignatureDateTime	datetime2	8	False
Operator	varchar(20)	20	False
ItemKey	varchar(150)	150	False
ComputerName	varchar(150)	150	True
ProgramName	varchar(100)	100	False
ConditionName	varchar(15)	15	True
AlreadyEntered	bit	1	True
BUYECDECLFACTAFTER	varchar(255)	255	True
BUYECDECLFACTBEFORE	varchar(255)	255	True
BUYECDECLRATEAFTER	varchar(255)	255	True
BUYECDECLRATEBEFORE	varchar(255)	255	True
BUYEXCHANGERATEAFTER	varchar(255)	255	True
BUYEXCHANGERATEBEFORE	varchar(255)	255	True
BUYMULDIVAFTER	varchar(255)	255	True
BUYMULDIVBEFORE	varchar(255)	255	True
DESCRIPTIONAFTER	varchar(255)	255	True
DESCRIPTIONBEFORE	varchar(255)	255	True
EUROAFTER	varchar(255)	255	True
EUROBEFORE	varchar(255)	255	True
FIXEDRATEAFTER	varchar(255)	255	True
FIXEDRATEBEFORE	varchar(255)	255	True
INTERMEDIATECURAFTER	varchar(255)	255	True
INTERMEDIATECURBEFORE	varchar(255)	255	True
SELLECDECLFACTAFTER	varchar(255)	255	True
SELLECDECLFACTBEFORE	varchar(255)	255	True
SELLECDECLRATEAFTER	varchar(255)	255	True
SELLECDECLRATEBEFORE	varchar(255)	255	True
SELLEXCHANGERATEAFTER	varchar(255)	255	True
SELLEXCHANGERATEBEFORE	varchar(255)	255	True
SELLMULDIVAFTER	varchar(255)	255	True
SELLMULDIVBEFORE	varchar(255)	255	True
TOLERANCEAFTER	varchar(255)	255	True

Author: Johnson, Chris

TOLERANCEBEFORE	varchar(255)	255	True
TRIANGREQDAFTER	varchar(255)	255	True
TRIANGREQDBEFORE	varchar(255)	255	True

SQL Script

```
CREATE TABLE [History].[ArchiveTblCurrency27OCT2016112011017]
(
[TransactionDescription] [varchar] (150) COLLATE Latin1_General_BIN NULL,
[DatabaseName] [varchar] (150) COLLATE Latin1_General_BIN NOT NULL,
[SignatureDateTime] [datetime2] NOT NULL,
[Operator] [varchar] (20) COLLATE Latin1_General_BIN NOT NULL,
[ItemKey] [varchar] (150) COLLATE Latin1_General_BIN NOT NULL,
[ComputerName] [varchar] (150) COLLATE Latin1_General_BIN NULL,
[ProgramName] [varchar] (100) COLLATE Latin1_General_BIN NOT NULL,
[ConditionName] [varchar] (15) COLLATE Latin1_General_BIN NULL,
[AlreadyEntered] [bit] NULL,
[BUYECDECLFACTAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[BUYECDECLFACTBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[BUYECDECLRATEAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[BUYECDECLRATEBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[BUYEXCHANGERATEAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[BUYEXCHANGERATEBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[BUYMULDIVAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[BUYMULDIVBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[DESCRIPTIONAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[DESCRIPTIONBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[EUROAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[EUROBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[FIXEDRATEAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[FIXEDRATEBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[INTERMEDIATECURAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[INTERMEDIATECURBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[SELLECDECLFACTAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[SELLECDECLFACTBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[SELLECDECLRATEAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[SELLECDECLRATEBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[SELLEXCHANGERATEAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[SELLEXCHANGERATEBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[SELLMULDIVAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[SELLMULDIVBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[TOLERANCEAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[TOLERANCEBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[TRIANGREQDAFTER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[TRIANGREQDBEFORE] [varchar] (255) COLLATE Latin1_General_BIN NULL
) ON [PRIMARY]
GO
EXEC sp_addextendedproperty N'MS_Description', N'Logs from TblCurrency change logs',
'SCHEMA', N'History', 'TABLE', N'ArchiveTblCurrency27OCT2016112011017', NULL, NULL
GO
```

Uses

[History]

Author: Johnson, Chris

[History].[ArchiveWipInspect27OCT2016112011087]

Columns

Name	Data Type	Max Length (Bytes)	Allow Nulls	Identity
WID	int	4	False	1 - 1
TransactionDescription	varchar(150)	150	True	
SignatureDatetime	datetime2	8	False	
Operator	varchar(20)	20	False	
ProgramName	varchar(20)	20	False	
Ranking	bigint	8	True	
ItemKey	varchar(150)	150	False	
DatabaseName	varchar(150)	150	False	
BINLOCATION	varchar(255)	255	True	
EXPIRYDATE	datetime	8	True	
INSPECTIONREFERENCE	varchar(255)	255	True	
JOB	varchar(255)	255	True	
LOTNUMBER	varchar(255)	255	True	
RELEASE	varchar(255)	255	True	
STOCKCODE	varchar(255)	255	True	
VERSION	varchar(255)	255	True	
WAREHOUSE	varchar(255)	255	True	
WIPINSPECTKEY	varchar(255)	255	True	
ALTERNATEWAREHOUSE	varchar(255)	255	True	
ALTWAREHOUSEFLAG	varchar(255)	255	True	
QUANTITY	float	8	True	

SQL Script

```
CREATE TABLE [History].[ArchiveWipInspect27OCT2016112011087]
(
[WID] [int] NOT NULL IDENTITY(1, 1),
[TransactionDescription] [varchar] (150) COLLATE Latin1_General_BIN NULL,
[SignatureDatetime] [datetime2] NOT NULL,
[Operator] [varchar] (20) COLLATE Latin1_General_BIN NOT NULL,
[ProgramName] [varchar] (20) COLLATE Latin1_General_BIN NOT NULL,
[Ranking] [bigint] NULL,
[ItemKey] [varchar] (150) COLLATE Latin1_General_BIN NOT NULL,
[DatabaseName] [varchar] (150) COLLATE Latin1_General_BIN NOT NULL,
[BINLOCATION] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[EXPIRYDATE] [datetime] NULL,
[INSPECTIONREFERENCE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[JOB] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[LOTNUMBER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[RELEASE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
```

Author: Johnson, Chris

```
[STOCKCODE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[VERSION] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[WAREHOUSE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[WIPINSPECTKEY] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[ALTERNATEWAREHOUSE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[ALTWAREHOUSEFLAG] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[QUANTITY] [float] NULL
) ON [PRIMARY]
GO
```

Uses

[History]

[History].[ArchiveWipJobAllLab27OCT2016112011113]

Columns

Name	Data Type	Max Length (Bytes)	Allow Nulls	Identity
WID	int	4	False	1 - 1
TransactionDescription	varchar(150)	150	True	
SignatureDatetime	datetime2	8	False	
Operator	varchar(20)	20	False	
ProgramName	varchar(20)	20	False	
Ranking	bigint	8	True	
ItemKey	varchar(150)	150	False	
DatabaseName	varchar(150)	150	False	
EMPLOYEE	varchar(255)	255	True	
ENTRYDATE	datetime	8	True	
INSPECTIONFLAG	varchar(255)	255	True	
JOB	varchar(255)	255	True	
MACHINE	varchar(255)	255	True	
OPERATION	float	8	True	
QUANTITYCOMPLETED	float	8	True	
WIPJOBALLLABKEY	varchar(255)	255	True	
WORKCENTER	varchar(255)	255	True	

SQL Script

```
CREATE TABLE [History].[ArchiveWipJobAllLab27OCT2016112011113]
(
[WID] [int] NOT NULL IDENTITY(1, 1),
[TransactionDescription] [varchar] (150) COLLATE Latin1_General_BIN NULL,
[SignatureDatetime] [datetime2] NOT NULL,
[Operator] [varchar] (20) COLLATE Latin1_General_BIN NOT NULL,
[ProgramName] [varchar] (20) COLLATE Latin1_General_BIN NOT NULL,
[Ranking] [bigint] NULL,
[ItemKey] [varchar] (150) COLLATE Latin1_General_BIN NOT NULL,
[DatabaseName] [varchar] (150) COLLATE Latin1_General_BIN NOT NULL,
[EMPLOYEE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[ENTRYDATE] [datetime] NULL,
[INSPECTIONFLAG] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[JOB] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[MACHINE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[OPERATION] [float] NULL,
[QUANTITYCOMPLETED] [float] NULL,
[WIPJOBALLLABKEY] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[WORKCENTER] [varchar] (255) COLLATE Latin1_General_BIN NULL
) ON [PRIMARY]
GO
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Tables> History.ArchiveWipJobAllLab27OCT2016112011113

Uses

[History]

Author: Johnson, Chris

[History].[ArchiveWipJobAllMat27OCT2016112011123]

Columns

Name	Data Type	Max Length (Bytes)	Allow Nulls	Identity
WID	int	4	False	1 - 1
TransactionDescription	varchar(150)	150	True	
SignatureDatetime	datetime2	8	False	
Operator	varchar(20)	20	False	
ProgramName	varchar(20)	20	False	
Ranking	bigint	8	True	
ItemKey	varchar(150)	150	False	
DatabaseName	varchar(150)	150	False	
JOB	varchar(255)	255	True	
OPERATORCODE	varchar(255)	255	True	
QUANTITY	float	8	True	
SOURCEAPPLICATION	varchar(255)	255	True	
STOCKCODE	varchar(255)	255	True	
WAREHOUSE	varchar(255)	255	True	
WIPJOBALLMATKEY	varchar(255)	255	True	

SQL Script

```
CREATE TABLE [History].[ArchiveWipJobAllMat27OCT2016112011123]
(
[WID] [int] NOT NULL IDENTITY(1, 1),
[TransactionDescription] [varchar] (150) COLLATE Latin1_General_BIN NULL,
[SignatureDatetime] [datetime2] NOT NULL,
[Operator] [varchar] (20) COLLATE Latin1_General_BIN NOT NULL,
[ProgramName] [varchar] (20) COLLATE Latin1_General_BIN NOT NULL,
[Ranking] [bigint] NULL,
[ItemKey] [varchar] (150) COLLATE Latin1_General_BIN NOT NULL,
[DatabaseName] [varchar] (150) COLLATE Latin1_General_BIN NOT NULL,
[JOB] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[OPERATORCODE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[QUANTITY] [float] NULL,
[SOURCEAPPLICATION] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[STOCKCODE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[WAREHOUSE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[WIPJOBALLMATKEY] [varchar] (255) COLLATE Latin1_General_BIN NULL
) ON [PRIMARY]
GO
```

Uses

[History]

Author: Johnson, Chris

[History].[ArchiveWipMaster27OCT2016112011140]

Columns

Name	Data Type	Max Length (Bytes)	Allow Nulls	Identity
WID	int	4	False	1 - 1
TransactionDescription	varchar(150)	150	True	
SignatureDatetime	datetime2	8	False	
Operator	varchar(20)	20	False	
ProgramName	varchar(20)	20	False	
Ranking	bigint	8	True	
ItemKey	varchar(150)	150	False	
DatabaseName	varchar(150)	150	False	
BINLOCATION	varchar(255)	255	True	
CONDITIONDESCRIPTION	varchar(255)	255	True	
COSTBASIS	varchar(255)	255	True	
CURRENTCOMPANYDATE	datetime	8	True	
CURRENTCOMPANYID	varchar(255)	255	True	
CURRENTCOMPANYNAME	varchar(255)	255	True	
CURRENTLABORVALUE	float	8	True	
CURRENTMATERIALVALUE	float	8	True	
CURRENTOPERATINGSYSTEMDATE	datetime	8	True	
CURRENTOPERATINGSYSTEMTIME	varchar(255)	255	True	
FIFO/LIFOCOST	float	8	True	
IMPORTFLAG	varchar(255)	255	True	
INSPECTIONQUANTITY	float	8	True	
INSPECTIONREFERENCE	varchar(255)	255	True	
JOB	varchar(255)	255	True	
JOBCOMPLETE	varchar(255)	255	True	
JOBTYPE	varchar(255)	255	True	
JOURNAL	float	8	True	
JOURNALPOSTINGMONTH	float	8	True	
JOURNALPOSTINGYEAR	float	8	True	
LANGUAGECODE	varchar(255)	255	True	
LOTNUMBER	varchar(255)	255	True	
MASTERJOB	varchar(255)	255	True	
MULTIPLEBINFLAG	varchar(255)	255	True	
OPERATORCODE	varchar(255)	255	True	
OPERATORCURRENTGROUPCODE	varchar(255)	255	True	
OPERATOREMAILADDRESS	varchar(255)	255	True	
OPERATORLOCATION	varchar(255)	255	True	
OPERATORNAME	varchar(255)	255	True	

Author: Johnson, Chris

OPERATORPRIMARYGROUPCODE	varchar(255)	255	True	
OUTSTANDINGQUANTITY	float	8	True	
QUANTITY	float	8	True	
QUANTITYPLUSINSPECTIONQUANTITY	float	8	True	
RECEIPTCOST	float	8	True	
RELEASE	varchar(255)	255	True	
SALESORDER	varchar(255)	255	True	
SALESORDERLINECOMPLETEFLAG	varchar(255)	255	True	
SALESORDERLINENUMBER	float	8	True	
SOURCEAPPLICATION	varchar(255)	255	True	
STOCKCODE	varchar(255)	255	True	
TOTALLABOREXPECTEDVALUE	float	8	True	
TOTALLABORISSUEDVALUE	float	8	True	
TOTALMATERIALEXPECTEDVALUE	float	8	True	
TOTALMATERIALISSUEDVALUE	float	8	True	
UPDATESALESORDERLINEORDERQUANTITY	varchar(255)	255	True	
VERSION	varchar(255)	255	True	
WAREHOUSE	varchar(255)	255	True	

SQL Script

```
CREATE TABLE [History].[ArchiveWipMaster27OCT2016112011140]
(
[WID] [int] NOT NULL IDENTITY(1, 1),
[TransactionDescription] [varchar] (150) COLLATE Latin1_General_BIN NULL,
[SignatureDatetime] [datetime2] NOT NULL,
[Operator] [varchar] (20) COLLATE Latin1_General_BIN NOT NULL,
[ProgramName] [varchar] (20) COLLATE Latin1_General_BIN NOT NULL,
[Ranking] [bigint] NULL,
[ItemKey] [varchar] (150) COLLATE Latin1_General_BIN NOT NULL,
[DatabaseName] [varchar] (150) COLLATE Latin1_General_BIN NOT NULL,
[BINLOCATION] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[CONDITIONDESCRIPTION] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[COSTBASIS] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[CURRENTCOMPANYDATE] [datetime] NULL,
[CURRENTCOMPANYID] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[CURRENTCOMPANYNAME] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[CURRENTLABORVALUE] [float] NULL,
[CURRENTMATERIALVALUE] [float] NULL,
[CURRENTOPERATINGSYSTEMDATE] [datetime] NULL,
[CURRENTOPERATINGSYSTEMTIME] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[FIFO/LIFOCOST] [float] NULL,
[IMPORTFLAG] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[INSPECTIONQUANTITY] [float] NULL,
[INSPECTIONREFERENCE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[JOB] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[JOBCOMPLETE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[JOBTYPE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[JOURNAL] [float] NULL,
[JOURNALPOSTINGMONTH] [float] NULL,
[JOURNALPOSTINGYEAR] [float] NULL,
```

Author: Johnson, Chris

```
[LANGUAGECODE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[LOTNUMBER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[MASTERJOB] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[MULTIPLEBINFLAG] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[OPERATORCODE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[OPERATORCURRENTGROUPCODE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[OPERATOREMAILADDRESS] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[OPERATORLOCATION] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[OPERATORNAME] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[OPERATORPRIMARYGROUPCODE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[OUTSTANDINGQUANTITY] [float] NULL,
[QUANTITY] [float] NULL,
[QUANTITYPLUSINSPECTIONQUANTITY] [float] NULL,
[RECEIPTCOST] [float] NULL,
[RELEASE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[SALESORDER] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[SALESORDERLINECOMPLETEFLAG] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[SALESORDERLINENUMBER] [float] NULL,
[SOURCEAPPLICATION] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[STOCKCODE] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[TOTALLABOREXPECTEDVALUE] [float] NULL,
[TOTALLABORISSUEDVALUE] [float] NULL,
[TOTALMATERIALEXPECTEDVALUE] [float] NULL,
[TOTALMATERIALISSUEDVALUE] [float] NULL,
[UPDATESALESORDERLINEORDERQUANTITY] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[VERSION] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[WAREHOUSE] [varchar] (255) COLLATE Latin1_General_BIN NULL
) ON [PRIMARY]
GO
```

Uses

[History]

Author: Johnson, Chris

[History].[RedTagLogs]

Columns

Key	Name	Data Type	Max Length (Bytes)	Allow Nulls	Identity	Default
	TagID	int	4	False	1 - 1	
	TagDatetime	datetime2	8	True		(getdate())
	StoredProcDb	varchar(255)	255	True		
	StoredProcSchema	varchar(255)	255	True		
	StoredProcName	varchar(255)	255	True		
FK	UsedByType	char(1)	1	True		
	UsedByName	varchar(500)	500	True		
	UsedByDb	varchar(255)	255	True		

Foreign Keys

Name	Columns
FK__RedTagLog__UsedB__0E19EC5B	UsedByType->[Lookups].[RedTagsUsedByType].[UsedByType]

SQL Script

```
CREATE TABLE [History].[RedTagLogs]
(
[TagID] [int] NOT NULL IDENTITY(1, 1),
[TagDatetime] [datetime2] NULL CONSTRAINT [DF__RedTagLog__TagDa__0D25C822] DEFAULT
(getdate()),
[StoredProcDb] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[StoredProcSchema] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[StoredProcName] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[UsedByType] [char] (1) COLLATE Latin1_General_BIN NULL,
[UsedByName] [varchar] (500) COLLATE Latin1_General_BIN NULL,
[UsedByDb] [varchar] (255) COLLATE Latin1_General_BIN NULL
) ON [PRIMARY]
GO
ALTER TABLE [History].[RedTagLogs] ADD CONSTRAINT [FK__RedTagLog__UsedB__0E19EC5B]
FOREIGN KEY ([UsedByType]) REFERENCES [Lookups].[RedTagsUsedByType] ([UsedByType])
GO
```

Uses

[Lookups].[RedTagsUsedByType]
[History]

Used By

[Process].[UspInsert_RedTagLogs]

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Tables> History.RedTagLogs

[Report].[UspResults_RedTagDetails]
[Report].[UspResults_ReportSysRefresh_RedTagLogs]
[Review].[UspResults_LogStats]
[Review].[UspResults_ProcsMostRun]
[Review].[UspResults_ReportsDevelopmentRun]
[Review].[UspResults_ReportsMostRun]

Author: Johnson, Chris

[Lookups].[AdmTransactionIDs]

MS_Description

AdmTransaction Descriptions for use in reports

Columns

Name	Data Type	Max Length (Bytes)	Allow Nulls
TransactionId	varchar(6)	6	True
TransactionDescription	varchar(150)	150	True

SQL Script

```
CREATE TABLE [Lookups].[AdmTransactionIDs]
(
[TransactionId] [varchar] (6) COLLATE Latin1_General_BIN NULL,
[TransactionDescription] [varchar] (150) COLLATE Latin1_General_BIN NULL
) ON [PRIMARY]
GO
EXEC sp_addextendedproperty N'MS_Description', N'AdmTransaction Descriptions for use in reports', 'SCHEMA', N'Lookups', 'TABLE', N'AdmTransactionIDs', NULL, NULL
GO
```

Uses

[Lookups]

Used By

[Review].[UspResults_LookupDuplicates]

Author: Johnson, Chris

[Lookups].[ApJnlDistribExpenseType]

MS_Description

ApJnlDistribExpenseType Descriptions for use in reports

Columns

Key	Name	Data Type	Max Length (Bytes)	Allow Nulls
PK C	ExpenseType	varchar(3)	3	False
	ExpenseTypeDesc	varchar(150)	150	True

Indexes

Key	Name	Columns	Unique
PK C	ExpenseTypeKey	ExpenseType	True

SQL Script

```
CREATE TABLE [Lookups].[ApJnlDistribExpenseType]
(
[ExpenseType] [varchar] (3) COLLATE Latin1_General_BIN NOT NULL,
[ExpenseTypeDesc] [varchar] (150) COLLATE Latin1_General_BIN NULL
) ON [PRIMARY]
GO
ALTER TABLE [Lookups].[ApJnlDistribExpenseType] ADD CONSTRAINT [ExpenseTypeKey]
PRIMARY KEY CLUSTERED  ([ExpenseType]) ON [PRIMARY]
GO
EXEC sp_addextendedproperty N'MS_Description', N'ApJnlDistribExpenseType
Descriptions for use in reports', 'SCHEMA', N'Lookups', 'TABLE', N'ApJnlDistrib-
ExpenseType', NULL, NULL
GO
```

Uses

[Lookups]

Used By

[Report].[UspResults_InvoiceRunVerify]
[Review].[UspResults_LookupDuplicates]

Author: Johnson, Chris

[Lookups].[ApSupplier]

MS_Description

List of all suppliers

Columns

Key	Name	Data Type	Max Length (Bytes)	Allow Nulls
	Company	varchar(150)	150	True
	Supplier	varchar(150)	150	True
	SupplierName	varchar(150)	150	True
	LastUpdated	datetime2	8	True
	ActivePOFlag	bit	1	True

Indexes

Name	Columns
IX_ApSupplier_ActivePOFlag	Supplier, SupplierName, ActivePOFlag

SQL Script

```
CREATE TABLE [Lookups].[ApSupplier]
(
[Company] [varchar] (150) COLLATE Latin1_General_BIN NULL,
[Supplier] [varchar] (150) COLLATE Latin1_General_BIN NULL,
[SupplierName] [varchar] (150) COLLATE Latin1_General_BIN NULL,
[LastUpdated] [datetime2] NULL,
[ActivePOFlag] [bit] NULL
) ON [PRIMARY]
GO
CREATE NONCLUSTERED INDEX [IX_ApSupplier_ActivePOFlag] ON [Lookups].[ApSupplier]
([ActivePOFlag]) INCLUDE ([Supplier], [SupplierName]) ON [PRIMARY]
GO
EXEC sp_addextendedproperty N'MS_Description', N'List of all suppliers', 'SCHEMA',
N'Lookups', 'TABLE', N'ApSupplier', NULL, NULL
GO
```

Uses

[Lookups]

Used By

[dbo].[vw_APSuppliersWithActivePO]
[Lookups].[vw_APSuppliersWithActivePO]
[Process].[UspUpdate_ApSupplier]

Author: Johnson, Chris

[Lookups].[ArInvoicePayTrnType]

MS_Description

ArInvoicePay TrnType Descriptions for use in reports

Columns

Name	Data Type	Max Length (Bytes)	Allow Nulls
TrnType	char(1)	1	True
TrnTypeDesc	varchar(250)	250	True
LastUpdated	datetime2	8	True

SQL Script

```
CREATE TABLE [Lookups].[ArInvoicePayTrnType]
(
[TrnType] [char] (1) COLLATE Latin1_General_BIN NULL,
[TrnTypeDesc] [varchar] (250) COLLATE Latin1_General_BIN NULL,
[LastUpdated] [datetime2] NULL
) ON [PRIMARY]
GO
EXEC sp_addextendedproperty N'MS_Description', N'ArInvoicePay TrnType Descriptions for use in reports', 'SCHEMA', N'Lookups', 'TABLE', N'ArInvoicePayTrnType', NULL, NULL
GO
```

Uses

[Lookups]

Used By

[Process].[UspUpdate_ArInvoicePayTrnType]
[Report].[UspResults_APDaysToPayment]
[Report].[UspResults_ARDaysToPayment]
[Review].[UspResults_LookupDuplicates]

Author: Johnson, Chris

[Lookups].[BankBalances]

MS_Description

History of BankBalances

Columns

Key	Name	Data Type	Max Length (Bytes)	Allow Nulls	Default
	DatabaseName	varchar(150)	150	True	
	CompanyName	varchar(150)	150	True	
	Bank	varchar(10)	10	True	
	BankDescription	varchar(150)	150	True	
	CashGlCode	varchar(150)	150	True	
	BankCurrency	char(3)	3	True	
	CurrentBalance	numeric(20,7)	13	True	
	StatementBalance	numeric(20,7)	13	True	
	OutStandingDeposits	numeric(20,7)	13	True	
	OutStandingWithdrawals	numeric(20,7)	13	True	
	PrevMonth1CurrentBalance	numeric(20,7)	13	True	
	PrevMonth1StatementBalance	numeric(20,7)	13	True	
	PrevMonth1OutStandingDeposits	numeric(20,7)	13	True	
	PrevMonth1OutStandingWithdrawals	numeric(20,7)	13	True	
	PrevMonth2CurrentBalance	numeric(20,7)	13	True	
	PrevMonth2StatementBalance	numeric(20,7)	13	True	
	PrevMonth2OutStandingDeposits	numeric(20,7)	13	True	
	PrevMonth2OutStandingWithdrawals	numeric(20,7)	13	True	
	DateOfBalance	date	3	True	
	DateTimeOfBalance	datetime2	8	True	(getdate())

Indexes

Name	Columns
IX_BankBalances_DateOfBalance	DateOfBalance

SQL Script

```
CREATE TABLE [Lookups].[BankBalances]
(
[DatabaseName] [varchar] (150) COLLATE Latin1_General_BIN NULL,
[CompanyName] [varchar] (150) COLLATE Latin1_General_BIN NULL,
[Bank] [varchar] (10) COLLATE Latin1_General_BIN NULL,
[BankDescription] [varchar] (150) COLLATE Latin1_General_BIN NULL,
[CashGlCode] [varchar] (150) COLLATE Latin1_General_BIN NULL,
```

Author: Johnson, Chris

```
[BankCurrency] [char] (3) COLLATE Latin1_General_BIN NULL,
[CurrentBalance] [numeric] (20, 7) NULL,
[StatementBalance] [numeric] (20, 7) NULL,
[OutStandingDeposits] [numeric] (20, 7) NULL,
[OutStandingWithdrawals] [numeric] (20, 7) NULL,
[PrevMonth1CurrentBalance] [numeric] (20, 7) NULL,
[PrevMonth1StatementBalance] [numeric] (20, 7) NULL,
[PrevMonth1OutStandingDeposits] [numeric] (20, 7) NULL,
[PrevMonth1OutStandingWithdrawals] [numeric] (20, 7) NULL,
[PrevMonth2CurrentBalance] [numeric] (20, 7) NULL,
[PrevMonth2StatementBalance] [numeric] (20, 7) NULL,
[PrevMonth2OutStandingDeposits] [numeric] (20, 7) NULL,
[PrevMonth2OutStandingWithdrawals] [numeric] (20, 7) NULL,
[DateOfBalance] [date] NULL,
[DateTimeOfBalance] [datetime2] NULL CONSTRAINT [DF__BankBalan__DateT__09D45A2B]
DEFAULT (getdate())
) ON [PRIMARY]
GO
CREATE NONCLUSTERED INDEX [IX_BankBalances_DateOfBalance] ON [Lookups].[Bank-
Balances] ([DateOfBalance]) ON [PRIMARY]
GO
EXEC sp_addextendedproperty N'MS_Description', N'History of BankBalances', 'SCHEMA',
N'Lookups', 'TABLE', N'BankBalances', NULL, NULL
GO
```

Uses

[Lookups]

Used By

[Process].[UspUpdate_BankBalances]
[Review].[UspResults_LookupDuplicates]

Author: Johnson, Chris

[Lookups].[Bin]

MS_Description

list of bins to be used in reports

Columns

Name	Data Type	Max Length (Bytes)	Allow Nulls
Company	varchar(150)	150	True
Bin	varchar(150)	150	True
LastUpdated	datetime2	8	True

SQL Script

```
CREATE TABLE [Lookups].[Bin]
(
[Company] [varchar] (150) COLLATE Latin1_General_BIN NULL,
[Bin] [varchar] (150) COLLATE Latin1_General_BIN NULL,
[LastUpdated] [datetime2] NULL
) ON [PRIMARY]
GO
EXEC sp_addextendedproperty N'MS_Description', N'list of bins to be used in
reports', 'SCHEMA', N'Lookups', 'TABLE', N'Bin', NULL, NULL
GO
```

Uses

[Lookups]

Used By

[Process].[UspUpdate_Bin]
[Report].[UspResults_StockMovements]
[Review].[UspResults_LookupDuplicates]

Author: Johnson, Chris

[Lookups].[BudgetType]

MS_Description

list of budget types to be used in reports

Columns

Name	Data Type	Max Length (Bytes)	Allow Nulls
BudgetType	char(1)	1	True
BudgetTypeDesc	varchar(250)	250	True
LastUpdated	datetime2	8	True

SQL Script

```
CREATE TABLE [Lookups].[BudgetType]
(
[BudgetType] [char] (1) COLLATE Latin1_General_BIN NULL,
[BudgetTypeDesc] [varchar] (250) COLLATE Latin1_General_BIN NULL,
[LastUpdated] [datetime2] NULL
) ON [PRIMARY]
GO
EXEC sp_addextendedproperty N'MS_Description', N'list of budget types to be used in reports', 'SCHEMA', N'Lookups', 'TABLE', N'BudgetType', NULL, NULL
GO
```

Uses

[Lookups]

Used By

[Process].[UspUpdate_BudgetType]
[Report].[Results_BudgetTest]
[Review].[UspResults_LookupDuplicates]

Author: Johnson, Chris

[Lookups].[Buyers]

MS_Description

list of all buyers

Columns

Name	Data Type	Max Length (Bytes)	Allow Nulls
Company	varchar(150)	150	True
BuyerName	varchar(150)	150	True
LastUpdated	datetime2	8	True

SQL Script

```
CREATE TABLE [Lookups].[Buyers]
(
[Company] [varchar] (150) COLLATE Latin1_General_BIN NULL,
[BuyerName] [varchar] (150) COLLATE Latin1_General_BIN NULL,
[LastUpdated] [datetime2] NULL
) ON [PRIMARY]
GO
EXEC sp_addextendedproperty N'MS_Description', N'list of all buyers', 'SCHEMA',
N'Lookups', 'TABLE', N'Buyers', NULL, NULL
GO
```

Uses

[Lookups]

Used By

[Lookups].[vw_DistinctBuyerList]
[Process].[UspUpdate_Buyers]

Author: Johnson, Chris

[Lookups].[CompanyNames]

MS_Description

List of company names used in reports

Columns

Key	Name	Data Type	Max Length (Bytes)	Allow Nulls
PK C	Company	varchar(150)	150	False
	CompanyName	varchar(250)	250	True
	ShortName	varchar(250)	250	True
	Currency	varchar(10)	10	True
PK C	LastUpdated	datetime2	8	False

Indexes

Key	Name	Columns	Unique
PK C	PK_CompanyLastUpdated	Company, LastUpdated	True

SQL Script

```
CREATE TABLE [Lookups].[CompanyNames]
(
[Company] [varchar] (150) COLLATE Latin1_General_BIN NOT NULL,
[CompanyName] [varchar] (250) COLLATE Latin1_General_BIN NULL,
[ShortName] [varchar] (250) COLLATE Latin1_General_BIN NULL,
[Currency] [varchar] (10) COLLATE Latin1_General_BIN NULL,
[LastUpdated] [datetime2] NOT NULL
) ON [PRIMARY]
GO
ALTER TABLE [Lookups].[CompanyNames] ADD CONSTRAINT [PK_CompanyLastUpdated] PRIMARY KEY CLUSTERED  ([Company], [LastUpdated]) ON [PRIMARY]
GO
EXEC sp_addextendedproperty N'MS_Description', N'List of company names used in reports', 'SCHEMA', N'Lookups', 'TABLE', N'CompanyNames', NULL, NULL
GO
```

Uses

[Lookups]

Used By

[Process].[UspUpdate_CompanyNames]
[Report].[UspResults_AccPayable_AgedInvoices]
[Report].[UspResults_AllBankBalances]
[Report].[UspResults_AllPosAndReqs]

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Tables> Lookups.CompanyNames

[Report].[UspResults_ApAgingInvoices]
[Report].[UspResults_APDaysToPayment]
[Report].[UspResults_ApInvoicesWithPaymentDetails]
[Report].[UspResults_ArCustomers]
[Report].[UspResults_ARDaysToPayment]
[Report].[UspResults_AvailableLots]
[Report].[UspResults_BudgetsActuals]
[Report].[UspResults_ClosingBalancesInterco]
[Report].[UspResults_CompaniesCurrencyLatestExchangeRates]
[Report].[UspResults_CompanyTransactions]
[Report].[UspResults_FixedAssets]
[Report].[UspResults_GeneralLedgerControlStats]
[Report].[UspResults_GenJournalDetails]
[Report].[UspResults_GenJournalEntries]
[Report].[UspResults_GenledgerJournalsGrouped]
[Report].[UspResults_GL_Codes]
[Report].[UspResults_GLMovements]
[Report].[UspResults_GLMovementsIntercoPL]
[Report].[UspResults_InventoryInInspection]
[Report].[UspResults_InventoryInspectionTimes]
[Report].[UspResults_InvoiceRunVerify]
[Report].[UspResults_JobHeader]
[Report].[UspResults_JobHeader_AllJobs]
[Report].[UspResults_JournalDetails]
[Report].[UspResults_Labour]
[Report].[UspResults_LabourDetails]
[Report].[UspResults_LabourLoggedPerISOWeek]
[Report].[UspResults_LotsRetesting]
[Report].[UspResults_LotTraceability]
[Report].[UspResults_LotTraceability_WithRuntimes]
[Report].[UspResults_PaymentRunVerify]
[Report].[UspResults_PosNoReqs]
[Report].[UspResults_PurchaseOrderChanges]
[Report].[UspResults_PurchaseOrderDetails]
[Report].[UspResults_PurchaseOrdersHistory]
[Report].[UspResults_PurchaseOrdersOpen]
[Report].[UspResults_PurchaseOrderWithHistory]
[Report].[UspResults_ReqsAfterInvoices]
[Report].[UspResults_RequisitionStatus]
[Report].[UspResults_RequisitionUsers]
[Report].[UspResults_SalesOrderKPI]
[Report].[UspResults_SalesOrderStats]
[Report].[UspResults_ScrapCosts]
[Report].[UspResults_StockDispatches]
[Report].[UspResults_StockLevels]
[Report].[UspResults_StockOutput]
[Report].[UspResults_StockOutputDispatches2]
[Report].[UspResults_StockWithRunTimes]
[Report].[UspResults_TrialBalance]
[Report].[UspResults_UnpaidGRNAssets]
[Review].[UspResults_CompanyDuplicates]
[Review].[UspResults_CompanyNamesMissing]
[Review].[UspResults_LookupDuplicates]
[Report].[UspResults_GLMovementsCenter]
[Report].[UspResults_MissingGroupMaps]
[Report].[UspResults_MissingRI2Maps]

Author: Johnson, Chris

[Lookups].[CurrencyRates]

MS_Description

history of currency rates

Columns

Name	Data Type	Max Length (Bytes)	Allow Nulls
StartDateTime	datetime	8	True
EndDateTime	datetime	8	True
Currency	varchar(10)	10	True
CADDivision	numeric(12,7)	9	True
CHFDivision	numeric(12,7)	9	True
EURDivision	numeric(12,7)	9	True
GBPDivision	numeric(12,7)	9	True
JPYDivision	numeric(12,7)	9	True
USDDivision	numeric(12,7)	9	True
CADMultiply	numeric(12,7)	9	True
CHFMultiply	numeric(12,7)	9	True
EURMultiply	numeric(12,7)	9	True
GBPMultiply	numeric(12,7)	9	True
JPYMultiply	numeric(12,7)	9	True
USDMultiply	numeric(12,7)	9	True
LastUpdated	datetime2	8	True

SQL Script

```
CREATE TABLE [Lookups].[CurrencyRates]
(
[StartDateTime] [datetime] NULL,
[EndDateTime] [datetime] NULL,
[Currency] [varchar] (10) COLLATE Latin1_General_BIN NULL,
[CADDivision] [numeric] (12, 7) NULL,
[CHFDivision] [numeric] (12, 7) NULL,
[EURDivision] [numeric] (12, 7) NULL,
[GBPDivision] [numeric] (12, 7) NULL,
[JPYDivision] [numeric] (12, 7) NULL,
[USDDivision] [numeric] (12, 7) NULL,
[CADMultiply] [numeric] (12, 7) NULL,
[CHFMultiply] [numeric] (12, 7) NULL,
[EURMultiply] [numeric] (12, 7) NULL,
[GBPMultiply] [numeric] (12, 7) NULL,
[JPYMultiply] [numeric] (12, 7) NULL,
[USDMultiply] [numeric] (12, 7) NULL,
[LastUpdated] [datetime2] NULL
) ON [PRIMARY]
GO
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Tables> Lookups.CurrencyRates

```
EXEC sp_addextendedproperty N'MS_Description', N'history of currency rates',
'SCHEMA', N'Lookups', 'TABLE', N'CurrencyRates', NULL, NULL
GO
```

Uses

[Lookups]

Used By

[Process].[UspUpdate_CurrencyRates]
[Report].[Results_AllCurrencyRates]
[Report].[UspResults_AllPosAndReqs]
[Report].[UspResults_BudgetsActuals]
[Report].[UspResults_ClosingBalancesInterco]
[Report].[UspResults_CompaniesCurrencyLatestExchangeRates]
[Review].[UspResults_CurrencyRatesMissing]
[Review].[UspResults_LookupDuplicates]

Author: Johnson, Chris

[Lookups].[GenJournalCtlJnlSource]

MS_Description

Description of GenJournal CtlJnlSource

Columns

Name	Data Type	Max Length (Bytes)	Allow Nulls
GenJournalCtlJnlSource	char(2)	2	True
GenJournalCtlJnlSourceDesc	varchar(250)	250	True
LastUpdated	datetime2	8	True

SQL Script

```
CREATE TABLE [Lookups].[GenJournalCtlJnlSource]
(
[GenJournalCtlJnlSource] [char] (2) COLLATE Latin1_General_BIN NULL,
[GenJournalCtlJnlSourceDesc] [varchar] (250) COLLATE Latin1_General_BIN NULL,
[LastUpdated] [datetime2] NULL
) ON [PRIMARY]
GO
EXEC sp_addextendedproperty N'MS_Description', N'Description of GenJournal CtlJnl-
Source', 'SCHEMA', N'Lookups', 'TABLE', N'GenJournalCtlJnlSource', NULL, NULL
GO
```

Uses

[Lookups]

Used By

[Process].[UspUpdate_GenJournalCtlJnlSource]
[Report].[UspResults_CompanyTransactions]
[Report].[UspResults_GenJournalEntries]
[Review].[UspResults_LookupDuplicates]

Author: Johnson, Chris

[Lookups].[GenJournalCtlSource]

Columns

Key	Name	Data Type	Max Length (Bytes)	Allow Nulls
PK C	Source	char(1)	1	False
	SourceDescription	varchar(100)	100	True

Indexes

Key	Name	Columns	Unique
PK C	PK__GenJourn__09FAC3774842BCD4	Source	True

SQL Script

```
CREATE TABLE [Lookups].[GenJournalCtlSource]
(
[Source] [char] (1) COLLATE Latin1_General_BIN NOT NULL,
[SourceDescription] [varchar] (100) COLLATE Latin1_General_BIN NULL
) ON [PRIMARY]
GO
ALTER TABLE [Lookups].[GenJournalCtlSource] ADD CONSTRAINT [PK__Gen-
Journ__09FAC3774842BCD4] PRIMARY KEY CLUSTERED  ([Source]) ON [PRIMARY]
GO
```

Uses

[Lookups]

Used By

[Report].[UspResults_CompanyTransactions]
[Report].[UspResults_GenJournalEntries]
[Review].[UspResults_LookupDuplicates]

Author: Johnson, Chris

[Lookups].[GenJournalDetailSource]

MS_Description

Description of GenJournal DetailSource

Columns

Key	Name	Data Type	Max Length (Bytes)	Allow Nulls
PK C	GJSource	varchar(3)	3	False
	GJSourceDetail	varchar(100)	100	True

Indexes

Key	Name	Columns	Unique
PK C	PK__GenJourn__8DE11BC548C2CC0B	GJSource	True

SQL Script

```
CREATE TABLE [Lookups].[GenJournalDetailSource]
(
[GJSource] [varchar] (3) COLLATE Latin1_General_BIN NOT NULL,
[GJSourceDetail] [varchar] (100) COLLATE Latin1_General_BIN NULL
) ON [PRIMARY]
GO
ALTER TABLE [Lookups].[GenJournalDetailSource] ADD CONSTRAINT [PK__Gen-
Journ__8DE11BC548C2CC0B] PRIMARY KEY CLUSTERED  ([GJSource]) ON [PRIMARY]
GO
EXEC sp_addextendedproperty N'MS_Description', N'Description of GenJournal Detail-
Source', 'SCHEMA', N'Lookups', 'TABLE', N'GenJournalDetailSource', NULL, NULL
GO
```

Uses

[Lookups]

Used By

[Review].[UspResults_LookupDuplicates]

Author: Johnson, Chris

[Lookups].[GenJournalType]

MS_Description

Description of GenJournal Type

Columns

Key	Name	Data Type	Max Length (Bytes)	Allow Nulls
PK C	TypeCode	varchar(5)	5	False
	TypeDetail	varchar(100)	100	True

Indexes

Key	Name	Columns	Unique
PK C	PK__GenJourn__3E1CDC7D515D7FEF	TypeCode	True

SQL Script

```
CREATE TABLE [Lookups].[GenJournalType]
(
[TypeCode] [varchar] (5) COLLATE Latin1_General_BIN NOT NULL,
[TypeDetail] [varchar] (100) COLLATE Latin1_General_BIN NULL
) ON [PRIMARY]
GO
ALTER TABLE [Lookups].[GenJournalType] ADD CONSTRAINT [PK__Gen-
Journ__3E1CDC7D515D7FEF] PRIMARY KEY CLUSTERED  ([TypeCode]) ON [PRIMARY]
GO
EXEC sp_addextendedproperty N'MS_Description', N'Description of GenJournal Type',
'SCHEMA', N'Lookups', 'TABLE', N'GenJournalType', NULL, NULL
GO
```

Uses

[Lookups]

Used By

[Review].[UspResults_LookupDuplicates]

Author: Johnson, Chris

[Lookups].[GenTransactionSource]

MS_Description

Description of Gen Transaction Source

Columns

Name	Data Type	Max Length (Bytes)	Allow Nulls
Source	char(2)	2	True
SourceDesc	varchar(250)	250	True
LastUpdated	datetime2	8	True

SQL Script

```
CREATE TABLE [Lookups].[GenTransactionSource]
(
[Source] [char] (2) COLLATE Latin1_General_BIN NULL,
[SourceDesc] [varchar] (250) COLLATE Latin1_General_BIN NULL,
[LastUpdated] [datetime2] NULL
) ON [PRIMARY]
GO
EXEC sp_addextendedproperty N'MS_Description', N'Description of Gen Transaction
Source', 'SCHEMA', N'Lookups', 'TABLE', N'GenTransactionSource', NULL, NULL
GO
```

Uses

[Lookups]

Used By

[Process].[UspUpdate_GenTransactionSource]
[Report].[Results_BudgetTest]
[Report].[UspResults_JournalDetails]
[Review].[UspResults_LookupDuplicates]

Author: Johnson, Chris

[Lookups].[GlAccountCode]

MS_Description

Description of GL AccountCode

Columns

Name	Data Type	Max Length (Bytes)	Allow Nulls
Company	varchar(150)	150	True
GlAccountCode	char(5)	5	True
GlAccountDescription	varchar(150)	150	True
LastUpdated	datetime2	8	True

SQL Script

```
CREATE TABLE [Lookups].[GlAccountCode]
(
[Company] [varchar] (150) COLLATE Latin1_General_BIN NULL,
[GlAccountCode] [char] (5) COLLATE Latin1_General_BIN NULL,
[GlAccountDescription] [varchar] (150) COLLATE Latin1_General_BIN NULL,
[LastUpdated] [datetime2] NULL
) ON [PRIMARY]
GO
EXEC sp_addextendedproperty N'MS_Description', N'Description of GL AccountCode',
'SCHEMA', N'Lookups', 'TABLE', N'GlAccountCode', NULL, NULL
GO
```

Uses

[Lookups]

Used By

[Process].[UspUpdate_GlAccountCode]

Author: Johnson, Chris

[Lookups].[GLAccountType]

MS_Description

Description of GL AccountType

Columns

Name	Data Type	Max Length (Bytes)	Allow Nulls
GLAccountType	varchar(10)	10	True
GLAccountTypeDesc	varchar(250)	250	True
LastUpdated	datetime2	8	True

SQL Script

```
CREATE TABLE [Lookups].[GLAccountType]
(
[GLAccountType] [varchar] (10) COLLATE Latin1_General_BIN NULL,
[GLAccountTypeDesc] [varchar] (250) COLLATE Latin1_General_BIN NULL,
[LastUpdated] [datetime2] NULL
) ON [PRIMARY]
GO
EXEC sp_addextendedproperty N'MS_Description', N'Description of GL AccountType',
'SCHEMA', N'Lookups', 'TABLE', N'GLAccountType', NULL, NULL
GO
```

Uses

[Lookups]

Used By

[Process].[UspUpdate_GLAccountType]
[Report].[UspResults_BudgetsActuals]
[Report].[UspResults_GLMovements]
[Report].[UspResults_TrialBalance]

Author: Johnson, Chris

[Lookups].[GlAnalysisCategory]

Columns

Name	Data Type	Max Length (Bytes)	Allow Nulls
Company	varchar(150)	150	True
GlAnalysisCategory	varchar(150)	150	True
LastUpdated	datetime2	8	True

SQL Script

```
CREATE TABLE [Lookups].[GlAnalysisCategory]
(
[Company] [varchar] (150) COLLATE Latin1_General_BIN NULL,
[GlAnalysisCategory] [varchar] (150) COLLATE Latin1_General_BIN NULL,
[LastUpdated] [datetime2] NULL
) ON [PRIMARY]
GO
```

Uses

[Lookups]

Used By

[Lookups].[vw_AnalysisCategoryDistinct]
[Process].[UspUpdate_GlAnalysisCategories]
[Review].[UspResults_LookupDuplicates]

Author: Johnson, Chris

[Lookups].[GlExpenseCode]

MS_Description

Gl Expense code descriptions

Columns

Name	Data Type	Max Length (Bytes)	Allow Nulls
Company	varchar(150)	150	True
GlExpenseCode	char(5)	5	True
GlExpenseDescription	varchar(150)	150	True
LastUpdated	datetime2	8	True

SQL Script

```
CREATE TABLE [Lookups].[GlExpenseCode]
(
[Company] [varchar] (150) COLLATE Latin1_General_BIN NULL,
[GlExpenseCode] [char] (5) COLLATE Latin1_General_BIN NULL,
[GlExpenseDescription] [varchar] (150) COLLATE Latin1_General_BIN NULL,
[LastUpdated] [datetime2] NULL
) ON [PRIMARY]
GO
EXEC sp_addextendedproperty N'MS_Description', N'Gl Expense code descriptions',
'SCHEMA', N'Lookups', 'TABLE', N'GlExpenseCode', NULL, NULL
GO
```

Uses

[Lookups]

Used By

[Process].[UspUpdate_GlExpenseCode]
[Review].[UspResults_LookupDuplicates]

Author: Johnson, Chris

[Lookups].[GLMapping]

MS_Description

GL Mapping as provided by finance 2015

Columns

Name	Data Type	Max Length (Bytes)	Allow Nulls
GlCode	varchar(150)	150	True
GlStart	char(3)	3	True
GlMid	char(5)	5	True
GlEnd	char(3)	3	True
Mid1	char(3)	3	True
Mid2	char(2)	2	True
Company	varchar(10)	10	True
GlDescription	varchar(150)	150	True
Mapping1	varchar(255)	255	True
Mapping2	varchar(255)	255	True
Mapping3	varchar(255)	255	True
Mapping4	varchar(255)	255	True
Mapping5	varchar(255)	255	True

SQL Script

```
CREATE TABLE [Lookups].[GLMapping]
(
[GlCode] [varchar] (150) COLLATE Latin1_General_BIN NULL,
[GlStart] [char] (3) COLLATE Latin1_General_BIN NULL,
[GlMid] [char] (5) COLLATE Latin1_General_BIN NULL,
[GlEnd] [char] (3) COLLATE Latin1_General_BIN NULL,
[Mid1] [char] (3) COLLATE Latin1_General_BIN NULL,
[Mid2] [char] (2) COLLATE Latin1_General_BIN NULL,
[Company] [varchar] (10) COLLATE Latin1_General_BIN NULL,
[GlDescription] [varchar] (150) COLLATE Latin1_General_BIN NULL,
[Mapping1] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[Mapping2] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[Mapping3] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[Mapping4] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[Mapping5] [varchar] (255) COLLATE Latin1_General_BIN NULL
) ON [PRIMARY]
GO
EXEC sp_addextendedproperty N'MS_Description', N'GL Mapping as provided by finance
2015', 'SCHEMA', N'Lookups', 'TABLE', N'GLMapping', NULL, NULL
GO
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Tables> Lookups.GLMapping

Uses

[Lookups]

Used By

[Report].[UspResults_GenledgerJournalsGrouped]
[Report].[UspResults_GenMasterMapping]
[Report].[UspResults_LabourGlCodes]

Author: Johnson, Chris

[Lookups].[HolidayDays]

MS_Description

list of non-working days used to calculate holidays

Columns

Key	Name	Data Type	Max Length (Bytes)	Allow Nulls
PK C	Country	varchar(150)	150	False
	HolidayDesc	varchar(200)	200	True
PK C	HolidayDate	date	3	False

Indexes

Key	Name	Columns	Unique
PK C	HD_PrimKey	Country, HolidayDate	True
	HolidayDays_Date	HolidayDate, Country	

SQL Script

```
CREATE TABLE [Lookups].[HolidayDays]
(
[Country] [varchar] (150) COLLATE Latin1_General_BIN NOT NULL,
[HolidayDesc] [varchar] (200) COLLATE Latin1_General_BIN NULL,
[HolidayDate] [date] NOT NULL
) ON [PRIMARY]
GO
ALTER TABLE [Lookups].[HolidayDays] ADD CONSTRAINT [HD_PrimKey] PRIMARY KEY
CLUSTERED  ([Country], [HolidayDate]) ON [PRIMARY]
GO
CREATE NONCLUSTERED INDEX [HolidayDays_Date] ON [Lookups].[HolidayDays] ([Holiday-
Date], [Country]) ON [PRIMARY]
GO
EXEC sp_addextendedproperty N'MS_Description', N'list of non-working days used to
calculate holidays', 'SCHEMA', N'Lookups', 'TABLE', N'HolidayDays', NULL, NULL
GO
```

Uses

[Lookups]

Used By

[Review].[UspResults_LookupDuplicates]
[Process].[Udf_WorkingDays]

Author: Johnson, Chris

[Lookups].[InvMaster_PartCategory]

MS_Description

InvMaster Part Category description

Columns

Name	Data Type	Max Length (Bytes)	Allow Nulls
PartCategoryCode	varchar(10)	10	True
PartCategoryDescription	varchar(150)	150	True
LastUpdated	datetime2	8	True

SQL Script

```
CREATE TABLE [Lookups].[InvMaster_PartCategory]
(
[PartCategoryCode] [varchar] (10) COLLATE Latin1_General_BIN NULL,
[PartCategoryDescription] [varchar] (150) COLLATE Latin1_General_BIN NULL,
[LastUpdated] [datetime2] NULL
) ON [PRIMARY]
GO
EXEC sp_addextendedproperty N'MS_Description', N'InvMaster Part Category
description', 'SCHEMA', N'Lookups', 'TABLE', N'InvMaster_PartCategory', NULL, NULL
GO
```

Uses

[Lookups]

Used By

[Process].[UspUpdate_InvMaster_PartCategory]
[Report].[UspResults_WipStockRequiredPerJob]
[Review].[UspResults_LookupDuplicates]

[Lookups].[JnlPostingType]

MS_Description

JnlPosting Type description

Columns

Key	Name	Data Type	Max Length (Bytes)	Allow Nulls
PK C	JnlPostingType	char(1)	1	False
	JnlPostingTypeDesc	varchar(100)	100	True

Indexes

Key	Name	Columns	Unique
PK C	PK__JnlPosti__67A396901C72751C	JnlPostingType	True

SQL Script

```
CREATE TABLE [Lookups].[JnlPostingType]
(
[JnlPostingType] [char] (1) COLLATE Latin1_General_BIN NOT NULL,
[JnlPostingTypeDesc] [varchar] (100) COLLATE Latin1_General_BIN NULL
) ON [PRIMARY]
GO
ALTER TABLE [Lookups].[JnlPostingType] ADD CONSTRAINT [PK__Jnl-
Posti__67A396901C72751C] PRIMARY KEY CLUSTERED  ([JnlPostingType]) ON [PRIMARY]
GO
EXEC sp_addextendedproperty N'MS_Description', N'JnlPosting Type description',
'SCHEMA', N'Lookups', 'TABLE', N'JnlPostingType', NULL, NULL
GO
```

Uses

[Lookups]

Used By

[Report].[UspResults_CompanyTransactions]
[Report].[UspResults_GenJournalEntries]
[Review].[UspResults_LookupDuplicates]

Author: Johnson, Chris

[Lookups].[JnlStatus]

MS_Description

JnlStatus Description

Columns

Key	Name	Data Type	Max Length (Bytes)	Allow Nulls
PK C	JnlStatus	char(1)	1	False
	JnlStatusDesc	varchar(100)	100	True

Indexes

Key	Name	Columns	Unique
PK C	PK__JnlStatu__5CAFD7D463EA2AD9	JnlStatus	True

SQL Script

```
CREATE TABLE [Lookups].[JnlStatus]
(
[JnlStatus] [char] (1) COLLATE Latin1_General_BIN NOT NULL,
[JnlStatusDesc] [varchar] (100) COLLATE Latin1_General_BIN NULL
) ON [PRIMARY]
GO
ALTER TABLE [Lookups].[JnlStatus] ADD CONSTRAINT [PK__JnlStatu__5CAFD7D463EA2AD9]
PRIMARY KEY CLUSTERED  ([JnlStatus]) ON [PRIMARY]
GO
EXEC sp_addextendedproperty N'MS_Description', N'JnlStatus Description', 'SCHEMA',
N'Lookups', 'TABLE', N'JnlStatus', NULL, NULL
GO
```

Uses

[Lookups]

Used By

[Report].[UspResults_CompanyTransactions]
[Report].[UspResults_GenJournalEntries]
[Review].[UspResults_LookupDuplicates]

Author: Johnson, Chris

[Lookups].[LedgerGroupMaps]

Columns

Key	Name	Data Type	Max Length (Bytes)	Allow Nulls
PK C	GlGroup	varchar(500)	500	False
	Map1	varchar(500)	500	True
	Map2	varchar(500)	500	True
	Map3	varchar(500)	500	True

Indexes

Key	Name	Columns	Unique
PK C	PK__LedgerGr__34CC5B12AF214404	GlGroup	True

SQL Script

```
CREATE TABLE [Lookups].[LedgerGroupMaps]
(
[GlGroup] [varchar] (500) COLLATE Latin1_General_BIN NOT NULL,
[Map1] [varchar] (500) COLLATE Latin1_General_BIN NULL,
[Map2] [varchar] (500) COLLATE Latin1_General_BIN NULL,
[Map3] [varchar] (500) COLLATE Latin1_General_BIN NULL
) ON [PRIMARY]
GO
ALTER TABLE [Lookups].[LedgerGroupMaps] ADD CONSTRAINT [PK__Ledger-
Gr__34CC5B12AF214404] PRIMARY KEY CLUSTERED  ([GlGroup]) ON [PRIMARY]
GO
```

Uses

[Lookups]

Used By

[Report].[UspResults_BudgetsActuals]
[Report].[UspResults_MissingGroupMaps]

Author: Johnson, Chris

[Lookups].[LotTransactionTrnType]

MS_Description

LotTransaction TrnType Description

Columns

Key	Name	Data Type	Max Length (Bytes)	Allow Nulls
PK C	TrnType	char(1)	1	False
	TrnTypeDescription	varchar(100)	100	True

Indexes

Key	Name	Columns	Unique
PK C	LTTrnType	TrnType	True

SQL Script

```
CREATE TABLE [Lookups].[LotTransactionTrnType]
(
[TrnType] [char] (1) COLLATE Latin1_General_BIN NOT NULL,
[TrnTypeDescription] [varchar] (100) COLLATE Latin1_General_BIN NULL
) ON [PRIMARY]
GO
ALTER TABLE [Lookups].[LotTransactionTrnType] ADD CONSTRAINT [LTTrnType] PRIMARY KEY
CLUSTERED  ([TrnType]) ON [PRIMARY]
GO
EXEC sp_addextendedproperty N'MS_Description', N'LotTransaction TrnType
Description', 'SCHEMA', N'Lookups', 'TABLE', N'LotTransactionTrnType', NULL, NULL
GO
```

Uses

[Lookups]

Used By

[Report].[UspResults_LotTraceability]
[Report].[UspResults_LotTraceability_WithRuntimes]
[Report].[UspResults_StockOutput]
[Review].[UspResults_LookupDuplicates]

Author: Johnson, Chris

[Lookups].[MCompleteFlag]

Columns

Name	Data Type	Max Length (Bytes)	Allow Nulls
Company	varchar(150)	150	True
MCompleteFlagCode	char(5)	5	True
MCompleteFlagDescription	varchar(150)	150	True
LastUpdated	datetime2	8	True

SQL Script

```
CREATE TABLE [Lookups].[MCompleteFlag]
(
[Company] [varchar] (150) COLLATE Latin1_General_BIN NULL,
[MCompleteFlagCode] [char] (5) COLLATE Latin1_General_BIN NULL,
[MCompleteFlagDescription] [varchar] (150) COLLATE Latin1_General_BIN NULL,
[LastUpdated] [datetime2] NULL
) ON [PRIMARY]
GO
```

Uses

[Lookups]

Used By

[Process].[UspUpdate_MCompleteFlag]
[Report].[UspResults_PurchaseOrders]
[Review].[UspResults_LookupDuplicates]

Author: Johnson, Chris

[Lookups].[PorLineType]

Columns

Key	Name	Data Type	Max Length (Bytes)	Allow Nulls
PK C	PorLineType	int	4	False
	PorLineTypeDesc	varchar(150)	150	True
PK C	LastUpdated	datetime2	8	False

Indexes

Key	Name	Columns	Unique
PK C	PK__PorLineT__67453B017BDF3668	PorLineType, LastUpdated	True

SQL Script

```
CREATE TABLE [Lookups].[PorLineType]
(
[PorLineType] [int] NOT NULL,
[PorLineTypeDesc] [varchar] (150) COLLATE Latin1_General_BIN NULL,
[LastUpdated] [datetime2] NOT NULL
) ON [PRIMARY]
GO
ALTER TABLE [Lookups].[PorLineType] ADD CONSTRAINT [PK__PorLineT__67453B017BDF3668]
PRIMARY KEY CLUSTERED  ([PorLineType], [LastUpdated] DESC) ON [PRIMARY]
GO
```

Uses

[Lookups]

Used By

[Process].[UspUpdate_PorLineType]
[Report].[UspResults_PurchaseOrderWithHistory]
[Review].[UspResults_LookupDuplicates]

Author: Johnson, Chris

[Lookups].[ProductClass]

Columns

Name	Data Type	Max Length (Bytes)	Allow Nulls
Company	varchar(150)	150	True
ProductClass	varchar(150)	150	True
ProductClassDescription	varchar(250)	250	True
LastUpdated	datetime2	8	True

SQL Script

```
CREATE TABLE [Lookups].[ProductClass]
(
[Company] [varchar] (150) COLLATE Latin1_General_BIN NULL,
[ProductClass] [varchar] (150) COLLATE Latin1_General_BIN NULL,
[ProductClassDescription] [varchar] (250) COLLATE Latin1_General_BIN NULL,
[LastUpdated] [datetime2] NULL
) ON [PRIMARY]
GO
```

Uses

[Lookups]

Used By

[Process].[UspUpdate_ProductClass]
[Report].[UspResults_AllPosAndReqs]
[Report].[UspResults_PurchaseOrders]
[Review].[UspResults_LookupDuplicates]

Author: Johnson, Chris

[Lookups].[ProformaDocTypes]

Columns

Key	Name	Data Type	Max Length (Bytes)	Allow Nulls	Default
PK C	DOCUMENTTYPE	nvarchar(10)	20	False	
PK C	DOCUMENTFORMAT	nvarchar(10)	20	False	
	DocName	nvarchar(150)	300	True	
	InsertedDate	datetime2	8	True	(getdate())

Indexes

Key	Name	Columns	Unique
PK C	PfDType	DOCUMENTTYPE, DOCUMENTFORMAT	True

SQL Script

```
CREATE TABLE [Lookups].[ProformaDocTypes]
(
[DOCUMENTTYPE] [nvarchar] (10) COLLATE Latin1_General_BIN NOT NULL,
[DOCUMENTFORMAT] [nvarchar] (10) COLLATE Latin1_General_BIN NOT NULL,
[DocName] [nvarchar] (150) COLLATE Latin1_General_BIN NULL,
[InsertedDate] [datetime2] NULL CONSTRAINT [DF__ProformaD__Inser__676A338E] DEFAULT
(getdate())
) ON [PRIMARY]
GO
ALTER TABLE [Lookups].[ProformaDocTypes] ADD CONSTRAINT [PfDType] PRIMARY KEY
CLUSTERED  ([DOCUMENTTYPE], [DOCUMENTFORMAT]) ON [PRIMARY]
GO
```

Uses

[Lookups]

Used By

[Report].[UspResults_SalesOrderStats]

Author: Johnson, Chris

[Lookups].[PurchaseOrderInvoiceMapping]

Columns

Name	Data Type	Max Length (Bytes)	Allow Nulls
Company	varchar(150)	150	True
Grn	varchar(20)	20	True
Invoice	varchar(20)	20	True
PurchaseOrder	varchar(20)	20	True
LastUpdated	datetime2	8	True

SQL Script

```
CREATE TABLE [Lookups].[PurchaseOrderInvoiceMapping]
(
[Company] [varchar] (150) COLLATE Latin1_General_BIN NULL,
[Grn] [varchar] (20) COLLATE Latin1_General_BIN NULL,
[Invoice] [varchar] (20) COLLATE Latin1_General_BIN NULL,
[PurchaseOrder] [varchar] (20) COLLATE Latin1_General_BIN NULL,
[LastUpdated] [datetime2] NULL
) ON [PRIMARY]
GO
```

Uses

[Lookups]

Used By

[Process].[UspUpdate_PurchaseOrderInvoiceMapping]
[Report].[UspResults_OutstandingPurchaseOrders]
[Report].[UspResults_PurchaseOrdersInvoices]

Author: Johnson, Chris

[Lookups].[PurchaseOrderStatus]

Columns

Name	Data Type	Max Length (Bytes)	Allow Nulls
Company	varchar(150)	150	True
OrderStatusCode	char(5)	5	True
OrderStatusDescription	varchar(150)	150	True
LastUpdated	datetime2	8	True

SQL Script

```
CREATE TABLE [Lookups].[PurchaseOrderStatus]
(
[Company] [varchar] (150) COLLATE Latin1_General_BIN NULL,
[OrderStatusCode] [char] (5) COLLATE Latin1_General_BIN NULL,
[OrderStatusDescription] [varchar] (150) COLLATE Latin1_General_BIN NULL,
[LastUpdated] [datetime2] NULL
) ON [PRIMARY]
GO
```

Uses

[Lookups]

Used By

[Process].[UspUpdate_InvMaster_PartCategory]
[Process].[UspUpdate_PurchaseOrderStatus]
[Report].[UspResults_AllPosAndReqs]
[Report].[UspResults_OutstandingPurchaseOrders]
[Report].[UspResults_PosNoReqs]
[Report].[UspResults_PurchaseOrderDetails]
[Report].[UspResults_PurchaseOrders]
[Report].[UspResults_PurchaseOrdersOpen]
[Report].[UspResults_PurchaseOrderWithHistory]
[Review].[UspResults_LookupDuplicates]

Author: Johnson, Chris

[Lookups].[PurchaseOrderTaxStatus]

Columns

Name	Data Type	Max Length (Bytes)	Allow Nulls
Company	varchar(150)	150	True
TaxStatusCode	char(5)	5	True
TaxStatusDescription	varchar(150)	150	True
LastUpdated	datetime2	8	True

SQL Script

```
CREATE TABLE [Lookups].[PurchaseOrderTaxStatus]
(
[Company] [varchar] (150) COLLATE Latin1_General_BIN NULL,
[TaxStatusCode] [char] (5) COLLATE Latin1_General_BIN NULL,
[TaxStatusDescription] [varchar] (150) COLLATE Latin1_General_BIN NULL,
[LastUpdated] [datetime2] NULL
) ON [PRIMARY]
GO
```

Uses

[Lookups]

Used By

[Process].[UspUpdate_PurchaseOrderTaxStatus]
[Report].[UspResults_PurchaseOrders]
[Review].[UspResults_LookupDuplicates]

Author: Johnson, Chris

[Lookups].[PurchaseOrderType]

Columns

Name	Data Type	Max Length (Bytes)	Allow Nulls
Company	varchar(150)	150	True
OrderTypeCode	char(5)	5	True
OrderTypeDescription	varchar(150)	150	True
LastUpdated	datetime2	8	True

SQL Script

```
CREATE TABLE [Lookups].[PurchaseOrderType]
(
[Company] [varchar] (150) COLLATE Latin1_General_BIN NULL,
[OrderTypeCode] [char] (5) COLLATE Latin1_General_BIN NULL,
[OrderTypeDescription] [varchar] (150) COLLATE Latin1_General_BIN NULL,
[LastUpdated] [datetime2] NULL
) ON [PRIMARY]
GO
```

Uses

[Lookups]

Used By

[Process].[UspUpdate_PurchaseOrderType]
[Report].[UspResults_PurchaseOrders]
[Review].[UspResults_LookupDuplicates]

Author: Johnson, Chris

[Lookups].[RedTagsUsedByType]

Columns

Key	Name	Data Type	Max Length (Bytes)	Allow Nulls
PK C	UsedByType	char(1)	1	False
	UsedByDescription	varchar(150)	150	True

Indexes

Key	Name	Columns	Unique
PK C	PK__RedTagsU__F1E9B4D55790E575	UsedByType	True

SQL Script

```
CREATE TABLE [Lookups].[RedTagsUsedByType]
(
[UsedByType] [char] (1) COLLATE Latin1_General_BIN NOT NULL,
[UsedByDescription] [varchar] (150) COLLATE Latin1_General_BIN NULL
) ON [PRIMARY]
GO
ALTER TABLE [Lookups].[RedTagsUsedByType] ADD CONSTRAINT [PK__RedTagsU__-
F1E9B4D55790E575] PRIMARY KEY CLUSTERED  ([UsedByType]) ON [PRIMARY]
GO
```

Uses

[Lookups]

Used By

[History].[ArchiveRedTagLogs27OCT2016112011100]
[History].[RedTagLogs]
[Process].[UspInsert_RedTagLogs]
[Report].[UspResults_RedTagDetails]
[Review].[UspResults_LogStats]
[Review].[UspResults_ProcsMostRun]
[Review].[UspResults_ReportsDevelopmentRun]
[Review].[UspResults_ReportsMostRun]

Author: Johnson, Chris

[Lookups].[ReportIndexUserMaps]

Columns

Key	Name	Data Type	Max Length (Bytes)	Allow Nulls
PK C	ReportIndex2	varchar(100)	100	False
PK C	Map	varchar(500)	500	False
PK C	IsSummary	bit	1	False

Indexes

Key	Name	Columns	Unique
PK C	dsf	ReportIndex2, Map, IsSummary	True

SQL Script

```
CREATE TABLE [Lookups].[ReportIndexUserMaps]
(
[ReportIndex2] [varchar] (100) COLLATE Latin1_General_BIN NOT NULL,
[Map] [varchar] (500) COLLATE Latin1_General_BIN NOT NULL,
[IsSummary] [bit] NOT NULL
) ON [PRIMARY]
GO
ALTER TABLE [Lookups].[ReportIndexUserMaps] ADD CONSTRAINT [dsf] PRIMARY KEY
CLUSTERED  ([ReportIndex2], [Map], [IsSummary]) ON [PRIMARY]
GO
```

Uses

[Lookups]

Used By

[Report].[UspResults_BudgetsActuals]
[Report].[UspResults_MissingRI2Maps]

Author: Johnson, Chris

[Lookups].[ReqBuyers]

Columns

Name	Data Type	Max Length (Bytes)	Allow Nulls
BuyerName	varchar(150)	150	True
LastUpdated	datetime2	8	True

SQL Script

```
CREATE TABLE [Lookups].[ReqBuyers]
(
[BuyerName] [varchar] (150) COLLATE Latin1_General_BIN NULL,
[LastUpdated] [datetime2] NULL
) ON [PRIMARY]
GO
```

Uses

[Lookups]

Used By

[Process].[UspUpdate_ReqBuyers]

Author: Johnson, Chris

[Lookups].[ReqnStatus]

Columns

Name	Data Type	Max Length (Bytes)	Allow Nulls
Company	varchar(150)	150	True
ReqnStatusCode	char(5)	5	True
ReqnStatusDescription	varchar(150)	150	True
LastUpdated	datetime2	8	True

SQL Script

```
CREATE TABLE [Lookups].[ReqnStatus]
(
[Company] [varchar] (150) COLLATE Latin1_General_BIN NULL,
[ReqnStatusCode] [char] (5) COLLATE Latin1_General_BIN NULL,
[ReqnStatusDescription] [varchar] (150) COLLATE Latin1_General_BIN NULL,
[LastUpdated] [datetime2] NULL
) ON [PRIMARY]
GO
```

Uses

[Lookups]

Used By

[Process].[UspUpdate_ReqnStatus]
[Report].[UspResults_AllPosAndReqs]
[Report].[UspResults_RequisitionStatus]
[Review].[UspResults_LookupDuplicates]

Author: Johnson, Chris

[Lookups].[SalesOrderDocumentType]

Columns

Name	Data Type	Max Length (Bytes)	Allow Nulls
DocumentType	varchar(10)	10	True
DocumentTypeDesc	varchar(250)	250	True
LastUpdated	datetime2	8	True

SQL Script

```
CREATE TABLE [Lookups].[SalesOrderDocumentType]
(
[DocumentType] [varchar] (10) COLLATE Latin1_General_BIN NULL,
[DocumentTypeDesc] [varchar] (250) COLLATE Latin1_General_BIN NULL,
[LastUpdated] [datetime2] NULL
) ON [PRIMARY]
GO
```

Uses

[Lookups]

Used By

[Process].[UspUpdate_SalesOrderDocumentType]
[Report].[UspResults_SalesOrderStats]

Author: Johnson, Chris

[Lookups].[SalesOrderLineType]

Columns

Name	Data Type	Max Length (Bytes)	Allow Nulls
Company	varchar(150)	150	True
LineTypeCode	char(5)	5	True
LineTypeDescription	varchar(150)	150	True
LastUpdated	datetime2	8	True

SQL Script

```
CREATE TABLE [Lookups].[SalesOrderLineType]
(
[Company] [varchar] (150) COLLATE Latin1_General_BIN NULL,
[LineTypeCode] [char] (5) COLLATE Latin1_General_BIN NULL,
[LineTypeDescription] [varchar] (150) COLLATE Latin1_General_BIN NULL,
[LastUpdated] [datetime2] NULL
) ON [PRIMARY]
GO
```

Uses

[Lookups]

Used By

[Process].[UspUpdate_SalesOrderLineType]
[Report].[UspResults_SalesOrders]
[Review].[UspResults_LookupDuplicates]

Author: Johnson, Chris

[Lookups].[SalesOrderStatus]

Columns

Name	Data Type	Max Length (Bytes)	Allow Nulls
Company	varchar(150)	150	True
OrderStatusCode	char(5)	5	True
OrderStatusDescription	varchar(150)	150	True
LastUpdated	datetime2	8	True

SQL Script

```
CREATE TABLE [Lookups].[SalesOrderStatus]
(
[Company] [varchar] (150) COLLATE Latin1_General_BIN NULL,
[OrderStatusCode] [char] (5) COLLATE Latin1_General_BIN NULL,
[OrderStatusDescription] [varchar] (150) COLLATE Latin1_General_BIN NULL,
[LastUpdated] [datetime2] NULL
) ON [PRIMARY]
GO
```

Uses

[Lookups]

Used By

[Process].[UspUpdate_SalesOrderStatus]
[Report].[UspResults_AllSalesOrders]
[Report].[UspResults_SalesOrders]
[Review].[UspResults_LookupDuplicates]

Author: Johnson, Chris

[Lookups].[StockCode]

Columns

Key	Name	Data Type	Max Length (Bytes)	Allow Nulls
	Company	varchar(150)	150	True
	StockCode	varchar(150)	150	True
	StockDescription	varchar(150)	150	True
	PartCategory	varchar(5)	5	True
	ActivePOFlag	bit	1	True
	LastUpdated	datetime2	8	True

Indexes

Name	Columns
IX_StockCode_PartCategory	Company, StockCode, StockDescription, PartCategory

SQL Script

```
CREATE TABLE [Lookups].[StockCode]
(
[Company] [varchar] (150) COLLATE Latin1_General_BIN NULL,
[StockCode] [varchar] (150) COLLATE Latin1_General_BIN NULL,
[StockDescription] [varchar] (150) COLLATE Latin1_General_BIN NULL,
[PartCategory] [varchar] (5) COLLATE Latin1_General_BIN NULL,
[ActivePOFlag] [bit] NULL,
[LastUpdated] [datetime2] NULL
) ON [PRIMARY]
GO
CREATE NONCLUSTERED INDEX [IX_StockCode_PartCategory] ON [Lookups].[StockCode]
([PartCategory]) INCLUDE ([Company], [StockCode], [StockDescription]) ON [PRIMARY]
GO
```

Uses

[Lookups]

Used By

[Lookups].[StockCodes_Bought]
[Lookups].[vw_StockCodes_Bought_ActivePO]
[Process].[UspUpdate_StockCode]
[Report].[UspResults_StockMovements]

Author: Johnson, Chris

[Lookups].[TrnTypeAmountModifier]

Columns

Name	Data Type	Max Length (Bytes)	Allow Nulls
Company	varchar(150)	150	True
TrnType	char(5)	5	True
AmountModifier	int	4	True
LastUpdated	datetime2	8	True

SQL Script

```
CREATE TABLE [Lookups].[TrnTypeAmountModifier]
(
[Company] [varchar] (150) COLLATE Latin1_General_BIN NULL,
[TrnType] [char] (5) COLLATE Latin1_General_BIN NULL,
[AmountModifier] [int] NULL,
[LastUpdated] [datetime2] NULL
) ON [PRIMARY]
GO
```

Uses

[Lookups]

Used By

[Process].[UspUpdate_TrnTypeModifier]
[Report].[UspResults_AvailableLots]
[Report].[UspResults_AvailableStock]
[Report].[UspResults_JobStockDetails]
[Report].[UspResults_LotTraceability]
[Report].[UspResults_LotTraceability_WithRuntimes]
[Report].[UspResults_StockDispatches]
[Report].[UspResults_StockMovements]
[Report].[UspResults_StockOutput]
[Report].[UspResults_StockOutputDispatches2]
[Report].[UspResults_StockWithRunTimes]
[Review].[UspResults_LookupDuplicates]

Author: Johnson, Chris

[Lookups].[Warehouse]

Columns

Name	Data Type	Max Length (Bytes)	Allow Nulls
Company	varchar(150)	150	True
Warehouse	varchar(150)	150	True
WarehouseDescription	varchar(200)	200	True
LastUpdated	datetime2	8	True

SQL Script

```
CREATE TABLE [Lookups].[Warehouse]
(
[Company] [varchar] (150) COLLATE Latin1_General_BIN NULL,
[Warehouse] [varchar] (150) COLLATE Latin1_General_BIN NULL,
[WarehouseDescription] [varchar] (200) COLLATE Latin1_General_BIN NULL,
[LastUpdated] [datetime2] NULL
) ON [PRIMARY]
GO
```

Uses

[Lookups]

Used By

[Process].[UspUpdate_Warehouse]
[Report].[UspResults_LotTraceability]
[Report].[UspResults_LotTraceability_WithRuntimes]
[Report].[UspResults_StockDispatches]
[Report].[UspResults_StockMovements]
[Report].[UspResults_StockOutput]
[Report].[UspResults_StockOutputDispatches2]
[Review].[UspResults_LookupDuplicates]

Author: Johnson, Chris

[Process].[GrnInvoiceDetails]

Columns

Name	Data Type	Max Length (Bytes)	Allow Nulls	Default
LoadDate	datetime2	8	True	
IsComplete	bit	1	True	((0))
Company	varchar(150)	150	True	

SQL Script

```
CREATE TABLE [Process].[GrnInvoiceDetails]
(
[LoadDate] [datetime2] NULL,
[IsComplete] [bit] NULL CONSTRAINT [DF__GrnInvoic__IsCom__0BBCA29D] DEFAULT ((0)),
[Company] [varchar] (150) COLLATE Latin1_General_BIN NULL
) ON [PRIMARY]
GO
```

Uses

[Process]

Used By

[Report].[UspResults_GrnInvoiceDetails]

Author: Johnson, Chris

[Process].[RefreshTableTimes]

Columns

Name	Data Type	Max Length (Bytes)	Allow Nulls
SchemaName	varchar(100)	100	True
TableName	varchar(100)	100	True
LastUpdated	datetime2	8	True
OldRowCount	int	4	True
NewRowCount	int	4	True
UpdateStart	datetime2	8	True
UpdateEnd	datetime2	8	True
OldColumnCount	int	4	True
NewColumnCount	int	4	True
SecondsToRun	int	4	True
MinutesToRun	int	4	True
HoursToRun	int	4	True

SQL Script

```
CREATE TABLE [Process].[RefreshTableTimes]
(
[SchemaName] [varchar] (100) COLLATE Latin1_General_BIN NULL,
[TableName] [varchar] (100) COLLATE Latin1_General_BIN NULL,
[LastUpdated] [datetime2] NULL,
[OldRowCount] [int] NULL,
[NewRowCount] [int] NULL,
[UpdateStart] [datetime2] NULL,
[UpdateEnd] [datetime2] NULL,
[OldColumnCount] [int] NULL,
[NewColumnCount] [int] NULL,
[SecondsToRun] [int] NULL,
[MinutesToRun] [int] NULL,
[HoursToRun] [int] NULL
) ON [PRIMARY]
GO
```

Uses

[Process]

Author: Johnson, Chris

[Process].[Status_WipSubJobStock]

Columns

Name	Data Type	Max Length (Bytes)	Allow Nulls	Default
Job	varchar(20)	20	True	
Company	varchar(150)	150	True	
StartTime	datetime2	8	True	
CompleteTime	datetime2	8	True	
IsComplete	bit	1	True	((0))

SQL Script

```
CREATE TABLE [Process].[Status_WipSubJobStock]
(
[Job] [varchar] (20) COLLATE Latin1_General_BIN NULL,
[Company] [varchar] (150) COLLATE Latin1_General_BIN NULL,
[StartTime] [datetime2] NULL,
[CompleteTime] [datetime2] NULL,
[IsComplete] [bit] NULL CONSTRAINT [DF__Status_Wi__IsCom__6A30C649] DEFAULT ((0))
) ON [PRIMARY]
GO
```

Uses

[Process]

Used By

[Report].[UspResults_WipSubJobStock]

Author: Johnson, Chris

[Process].[SysproTransactionsLogged]

Columns

Key	Name	Data Type	Computed	Max Length (Bytes)	Allow Nulls	Default
(2)	TransactionDescription	varchar(150)		150	True	
PK (3)	DatabaseName	varchar(150)		150	False	
PK	SignatureDate	date		3	False	
PK	SignatureTime	int		4	False	
PK (2)	Operator	varchar(20)		20	False	
PK (3)	VariableDesc	varchar(100)		100	False	
PK (3)	ItemKey	varchar(150)		150	False	
	VariableType	char(1)		1	True	
	VarAlphaValue	varchar(255)		255	True	
	VarNumericValue	float		8	True	
	VarDateValue	datetime2		8	True	
(2)	ComputerName	varchar(150)		150	True	
PK (2)	ProgramName	varchar(100)		100	False	
PK (3)	TableName	varchar(150)		150	False	
(2)	ConditionName	varchar(15)		15	True	
(3)	AlreadyEntered	bit		1	True	((0))
(2)	IsError	bit		1	True	((0))
(3)	SignatureDateTime	datetime	True	8	True	

Computed columns

Name	Column definition
SignatureDateTime	(dateadd(millisecond,CONVERT([int],substring(right('0'+CONVERT([varchar](10),[Signature-Time],0),(8)),(7),(2)),0),dateadd(second,CONVERT([int],substring(right('0'+CONVERT([varchar](10),[Signature-Time],0),(8)),(5),(2)),0),dateadd(minute,CONVERT([int],substring(right('0'+CONVERT([varchar](10),[Signature-Time],0),(8)),(3),(2)),0),dateadd(hour,CONVERT([int],substring(right('0'+CONVERT([varchar](10),[SignatureTime],0),(8)),(1),(2)),0),CONVERT([datetime],[SignatureDate],0))))))

Indexes

Key	Name	Columns	Unique

Author: Johnson, Chris

PK	TDR_AllKeys	DatabaseName, SignatureDate, SignatureTime, ItemKey, Operator, ProgramName, VariableDesc, TableName	True
	IX_SysproTransactionsLogged_DatabaseName_Operator	TransactionDescription, SignatureDate, SignatureTime, VariableDesc, ItemKey, VariableType, VarAlphaValue, VarNumericValue, VarDateValue, ComputerName, ProgramName, TableName, ConditionName, AlreadyEntered, IsError, SignatureDateTime, DatabaseName, Operator	
	IX_SysproTransactionsLogged_DatabaseName_SignatureDateTime_VariableDesc_ItemKey	DatabaseName, SignatureDateTime, VariableDesc, ItemKey	
	IX_SysproTransactionsLogged_TableName	AlreadyEntered, IsError, TableName	
	IX_SysproTransactionsLogged_TableName_AlreadyEntered	ComputerName, ConditionName, DatabaseName, ItemKey, Operator, ProgramName, SignatureDateTime, TransactionDescription, VariableDesc, TableName, AlreadyEntered	

SQL Script

```
CREATE TABLE [Process].[SysproTransactionsLogged]
(
[TransactionDescription] [varchar] (150) COLLATE Latin1_General_BIN NULL,
```

Author: Johnson, Chris

```
[DatabaseName] [varchar] (150) COLLATE Latin1_General_BIN NOT NULL,
[SignatureDate] [date] NOT NULL,
[SignatureTime] [int] NOT NULL,
[Operator] [varchar] (20) COLLATE Latin1_General_BIN NOT NULL,
[VariableDesc] [varchar] (100) COLLATE Latin1_General_BIN NOT NULL,
[ItemKey] [varchar] (150) COLLATE Latin1_General_BIN NOT NULL,
[VariableType] [char] (1) COLLATE Latin1_General_BIN NULL,
[VarAlphaValue] [varchar] (255) COLLATE Latin1_General_BIN NULL,
[VarNumericValue] [float] NULL,
[VarDateValue] [datetime2] NULL,
[ComputerName] [varchar] (150) COLLATE Latin1_General_BIN NULL,
[ProgramName] [varchar] (100) COLLATE Latin1_General_BIN NOT NULL,
[TableName] [varchar] (150) COLLATE Latin1_General_BIN NOT NULL,
[ConditionName] [varchar] (15) COLLATE Latin1_General_BIN NULL,
[AlreadyEntered] [bit] NULL CONSTRAINT [DF__SysproTra__Alrea__3587F3E0] DEFAULT
((0)),
[IsError] [bit] NULL CONSTRAINT [DF__SysproTra__IsErr__18427513] DEFAULT ((0)),
[SignatureDateTime] AS
(dateadd(millisecond,CONVERT([int],substring(right('0'+CONVERT([varchar](10),[Signat
ure-
Time],0),(8)),(7),(2)),0),dateadd(second,CONVERT([int],substring(right('0'+CONVERT([
varchar](10),[Signature-
Time],0),(8)),(5),(2)),0),dateadd(minute,CONVERT([int],substring(right('0'+CONVERT([
varchar](10),[Signature-
Time],0),(8)),(3),(2)),0),dateadd(hour,CONVERT([int],substring(right('0'+CONVERT([va
rchar](10),[SignatureTime],0),(8)),(1),(2)),0),CONVERT([datetime],[Signature-
Date],0))))))
) ON [PRIMARY]
GO
ALTER TABLE [Process].[SysproTransactionsLogged] ADD CONSTRAINT [TDR_AllKeys]
PRIMARY KEY NONCLUSTERED  ([DatabaseName], [SignatureDate], [SignatureTime], [Item-
Key], [Operator], [ProgramName], [VariableDesc], [TableName]) WITH (IGNORE_DUP_-
KEY=ON) ON [PRIMARY]
GO
CREATE NONCLUSTERED INDEX [IX_SysproTransactionsLogged_DatabaseName_Operator] ON
[Process].[SysproTransactionsLogged] ([DatabaseName], [Operator]) INCLUDE ([Already-
Entered], [ComputerName], [ConditionName], [IsError], [ItemKey], [ProgramName],
[SignatureDate], [SignatureDateTime], [SignatureTime], [TableName], [Transaction-
Description], [VarAlphaValue], [VarDateValue], [VariableDesc], [VariableType], [Var-
NumericValue]) ON [PRIMARY]
GO
CREATE NONCLUSTERED INDEX [IX_SysproTransactionsLogged_DatabaseName_SignatureDate-
Time_VariableDesc_ItemKey] ON [Process].[SysproTransactionsLogged] ([DatabaseName],
[SignatureDateTime], [VariableDesc], [ItemKey]) ON [PRIMARY]
GO
CREATE NONCLUSTERED INDEX [IX_SysproTransactionsLogged_TableName] ON
[Process].[SysproTransactionsLogged] ([TableName]) INCLUDE ([AlreadyEntered], [Is-
Error]) ON [PRIMARY]
GO
CREATE NONCLUSTERED INDEX [IX_SysproTransactionsLogged_TableName_AlreadyEntered] ON
[Process].[SysproTransactionsLogged] ([TableName], [AlreadyEntered]) INCLUDE
([ComputerName], [ConditionName], [DatabaseName], [ItemKey], [Operator], [Program-
Name], [SignatureDateTime], [TransactionDescription], [VariableDesc]) ON [PRIMARY]
GO
```

Uses

[Process]

Used By

[Process].[UspAlter_CheckAndAddColumns]

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Tables> Process.SysproTransactionsLogged

[Process].[UspPopulate_HistoryTables1]
[Process].[UspPopulate_UnpivotHistory]
[Review].[UspResults_SysproTransactionsCheck]
[Review].[UspResults_SysproTransactionsTrending]

Author: Johnson, Chris

[Report].[GrnInvoiceDetails]

Columns

Key	Name	Data Type	Max Length (Bytes)	Allow Nulls
	DatabaseName	varchar(150)	150	True
	Supplier	varchar(150)	150	True
	Grn	varchar(50)	50	True
	TransactionType	varchar(5)	5	True
	Journal	int	4	True
	EntryNumber	int	4	True
	Invoice	varchar(150)	150	True
	PurchaseOrder	varchar(150)	150	True
	PurchaseOrderLine	int	4	True
	Requisition	varchar(150)	150	True
	RequisitionLine	int	4	True
	GlCode	varchar(50)	50	True
	Description	varchar(150)	150	True
	MatchedValue	decimal(15,3)	9	True
	MatchedDate	date	3	True
	StockCode	varchar(50)	50	True
	QtyReceived	decimal(20,12)	13	True
	MatchedYear	int	4	True
	MatchedMonth	int	4	True
	MatchedQty	decimal(20,12)	13	True
	Operator	varchar(150)	150	True
	Approver	varchar(150)	150	True
	OrigReceiptDate	date	3	True
	LoadDate	datetime2	8	True

Indexes

Name	Columns
IX_GrnInvoiceDetails_LoadDate_DatabaseName	Supplier, Grn, TransactionType, Journal, EntryNumber, Invoice, PurchaseOrder, PurchaseOrderLine, Requisition, RequisitionLine, GlCode, Description, MatchedValue, MatchedDate, StockCode, QtyReceived, MatchedYear, MatchedMonth, MatchedQty, Operator, Approver, OrigReceiptDate, LoadDate, DatabaseName

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Tables> Report.GrnInvoiceDetails

SQL Script

```
CREATE TABLE [Report].[GrnInvoiceDetails]
(
[DatabaseName] [varchar] (150) COLLATE Latin1_General_BIN NULL,
[Supplier] [varchar] (150) COLLATE Latin1_General_BIN NULL,
[Grn] [varchar] (50) COLLATE Latin1_General_BIN NULL,
[TransactionType] [varchar] (5) COLLATE Latin1_General_BIN NULL,
[Journal] [int] NULL,
[EntryNumber] [int] NULL,
[Invoice] [varchar] (150) COLLATE Latin1_General_BIN NULL,
[PurchaseOrder] [varchar] (150) COLLATE Latin1_General_BIN NULL,
[PurchaseOrderLine] [int] NULL,
[Requisition] [varchar] (150) COLLATE Latin1_General_BIN NULL,
[RequisitionLine] [int] NULL,
[GlCode] [varchar] (50) COLLATE Latin1_General_BIN NULL,
[Description] [varchar] (150) COLLATE Latin1_General_BIN NULL,
[MatchedValue] [decimal] (15, 3) NULL,
[MatchedDate] [date] NULL,
[StockCode] [varchar] (50) COLLATE Latin1_General_BIN NULL,
[QtyReceived] [decimal] (20, 12) NULL,
[MatchedYear] [int] NULL,
[MatchedMonth] [int] NULL,
[MatchedQty] [decimal] (20, 12) NULL,
[Operator] [varchar] (150) COLLATE Latin1_General_BIN NULL,
[Approver] [varchar] (150) COLLATE Latin1_General_BIN NULL,
[OrigReceiptDate] [date] NULL,
[LoadDate] [datetime2] NULL
) ON [PRIMARY]
GO
CREATE NONCLUSTERED INDEX [IX_GrnInvoiceDetails_LoadDate_DatabaseName] ON
[Report].[GrnInvoiceDetails] ([LoadDate], [DatabaseName]) INCLUDE ([Approver],
[Description], [EntryNumber], [GlCode], [Grn], [Invoice], [Journal], [MatchedDate],
[MatchedMonth], [MatchedQty], [MatchedValue], [MatchedYear], [Operator], [Orig-
ReceiptDate], [PurchaseOrder], [PurchaseOrderLine], [QtyReceived], [Requisition],
[RequisitionLine], [StockCode], [Supplier], [TransactionType]) ON [PRIMARY]
GO
```

Uses

[Report]

Used By

[Report].[UspResults_GrnInvoiceDetails]

Author: Johnson, Chris

[Report].[Results_WipSubJobStock]

Columns

Key	Name	Data Type	Max Length (Bytes)	Allow Nulls
	StartTime	datetime2	8	True
	Job	varchar(20)	20	True
	SubJob	varchar(20)	20	True
	SubJobDescription	varchar(50)	50	True
	SubJobUom	varchar(10)	10	True
	SequenceNum	varchar(6)	6	True
	SubJobQty	numeric(20,8)	13	True
	StockCode	varchar(20)	20	True
	StockDescription	varchar(50)	50	True
	UnitQtyReqdEnt	numeric(20,8)	13	True
	QtyIssuedEnt	numeric(20,8)	13	True
	FixedQtyPerFlag	char(1)	1	True
	Uom	varchar(10)	10	True
	AllocCompleted	char(1)	1	True
	OperationOffset	int	4	True
	WorkCentre	varchar(20)	20	True
	SubJobQtyTotal	numeric(20,8)	13	True
	IssMultLotsFlag	char(1)	1	True
	ReservedLotSerFlag	char(1)	1	True
	ReservedLotQty	numeric(20,8)	13	True
	ReservedLot	varchar(50)	50	True
	ReservedLotBin	varchar(20)	20	True
	ReservedLotWarehouse	varchar(20)	20	True
	ReservedLotQtyReserved	numeric(20,8)	13	True
	ReservedLotQtyIssued	numeric(20,8)	13	True
	AvailableLot	varchar(50)	50	True
	AvailableLotBin	varchar(20)	20	True
	AvailableLotWarehouse	varchar(20)	20	True
	AvailableLotQtyOnHand	numeric(20,8)	13	True
	AvailableLotExpiryDate	datetime2	8	True
	AvailableLotCreationDate	datetime2	8	True

Author: Johnson, Chris

	ReservedLotBleedNumber	varchar(20)	20	True
	ReservedLotDonorNumber	varchar(20)	20	True
	ReservedLotVendorBatchNumber	varchar(50)	50	True
	ReservedLotOldLotNumber	varchar(20)	20	True
	ReservedLotBleedDate	varchar(15)	15	True
	StockDecimals	int	4	True

Indexes

Name	Columns
IX_Results_WipSubJobStock_StartTime	Job, SubJob, SubJobDescription, SubJobUom, SequenceNum, SubJobQty, StockCode, StockDescription, UnitQtyReqdEnt, QtyIssuedEnt, FixedQtyPerFlag, Uom, AllocCompleted, OperationOffset, WorkCentre, SubJobQtyTotal, IssMultLotsFlag, ReservedLotSerFlag, ReservedLotQty, ReservedLot, ReservedLotBin, ReservedLotWarehouse, ReservedLotQtyReserved, ReservedLotQtyIssued, AvailableLot, AvailableLotBin, AvailableLotWarehouse, AvailableLotQtyOnHand, AvailableLotExpiryDate, AvailableLotCreationDate, ReservedLotBleedNumber, ReservedLotDonorNumber, ReservedLotVendorBatchNumber, ReservedLotOldLotNumber, ReservedLotBleedDate, StartTime

SQL Script

```
CREATE TABLE [Report].[Results_WipSubJobStock]
(
[StartTime] [datetime2] NULL,
[Job] [varchar] (20) COLLATE Latin1_General_BIN NULL,
[SubJob] [varchar] (20) COLLATE Latin1_General_BIN NULL,
[SubJobDescription] [varchar] (50) COLLATE Latin1_General_BIN NULL,
[SubJobUom] [varchar] (10) COLLATE Latin1_General_BIN NULL,
[SequenceNum] [varchar] (6) COLLATE Latin1_General_BIN NULL,
[SubJobQty] [numeric] (20, 8) NULL,
[StockCode] [varchar] (20) COLLATE Latin1_General_BIN NULL,
[StockDescription] [varchar] (50) COLLATE Latin1_General_BIN NULL,
[UnitQtyReqdEnt] [numeric] (20, 8) NULL,
[QtyIssuedEnt] [numeric] (20, 8) NULL,
[FixedQtyPerFlag] [char] (1) COLLATE Latin1_General_BIN NULL,
[Uom] [varchar] (10) COLLATE Latin1_General_BIN NULL,
[AllocCompleted] [char] (1) COLLATE Latin1_General_BIN NULL,
[OperationOffset] [int] NULL,
[WorkCentre] [varchar] (20) COLLATE Latin1_General_BIN NULL,
[SubJobQtyTotal] [numeric] (20, 8) NULL,
[IssMultLotsFlag] [char] (1) COLLATE Latin1_General_BIN NULL,
[ReservedLotSerFlag] [char] (1) COLLATE Latin1_General_BIN NULL,
[ReservedLotQty] [numeric] (20, 8) NULL,
[ReservedLot] [varchar] (50) COLLATE Latin1_General_BIN NULL,
[ReservedLotBin] [varchar] (20) COLLATE Latin1_General_BIN NULL,
[ReservedLotWarehouse] [varchar] (20) COLLATE Latin1_General_BIN NULL,
[ReservedLotQtyReserved] [numeric] (20, 8) NULL,
[ReservedLotQtyIssued] [numeric] (20, 8) NULL,
[AvailableLot] [varchar] (50) COLLATE Latin1_General_BIN NULL,
[AvailableLotBin] [varchar] (20) COLLATE Latin1_General_BIN NULL,
[AvailableLotWarehouse] [varchar] (20) COLLATE Latin1_General_BIN NULL,
```

Author: Johnson, Chris

```
[AvailableLotQtyOnHand] [numeric] (20, 8) NULL,
[AvailableLotExpiryDate] [datetime2] NULL,
[AvailableLotCreationDate] [datetime2] NULL,
[ReservedLotBleedNumber] [varchar] (20) COLLATE Latin1_General_BIN NULL,
[ReservedLotDonorNumber] [varchar] (20) COLLATE Latin1_General_BIN NULL,
[ReservedLotVendorBatchNumber] [varchar] (50) COLLATE Latin1_General_BIN NULL,
[ReservedLotOldLotNumber] [varchar] (20) COLLATE Latin1_General_BIN NULL,
[ReservedLotBleedDate] [varchar] (15) COLLATE Latin1_General_BIN NULL,
[StockDecimals] [int] NULL
) ON [PRIMARY]
GO
CREATE NONCLUSTERED INDEX [IX_Results_WipSubJobStock_StartTime] ON
[Report].[Results_WipSubJobStock] ([StartTime]) INCLUDE ([AllocCompleted],
[AvailableLot], [AvailableLotBin], [AvailableLotCreationDate], [AvailableLotExpiry-
Date], [AvailableLotQtyOnHand], [AvailableLotWarehouse], [FixedQtyPerFlag], [IssMult-
LotsFlag], [Job], [OperationOffset], [QtyIssuedEnt], [ReservedLot], [ReservedLot-
Bin], [ReservedLotBleedDate], [ReservedLotBleedNumber], [ReservedLotDonorNumber],
[ReservedLotOldLotNumber], [ReservedLotQty], [ReservedLotQtyIssued], [ReservedLotQty-
Reserved], [ReservedLotSerFlag], [ReservedLotVendorBatchNumber], [ReservedLot-
Warehouse], [SequenceNum], [StockCode], [StockDescription], [SubJob], [SubJob-
Description], [SubJobQty], [SubJobQtyTotal], [SubJobUom], [UnitQtyReqdEnt], [Uom],
[WorkCentre]) ON [PRIMARY]
GO
```

Uses

[Report]

Used By

[Report].[UspResults_WipSubJobStock]

Views

Objects

Name
dbo.vw_APSuppliersWithActivePO limit of Ap suppliers for use in look ups
Lookups.StockCodes_Bought
Lookups.vw_AnalysisCategoryDistinct
Lookups.vw_APSuppliersWithActivePO limit of Ap suppliers for use in look ups
Lookups.vw_DistinctBuyerList limit of Buyers for use in look ups
Lookups.vw_StockCodes_Bought_ActivePO limit of StockCodes for use in look ups

[dbo].[vw_APSuppliersWithActivePO]

MS_Description

limit of Ap suppliers for use in look ups

Columns

Name
Supplier
SupplierName

SQL Script

```
Create View [dbo].[vw_APSuppliersWithActivePO]
as
Select Distinct [AS].[Supplier]
     , [AS].[SupplierName]
FROM [Lookups].[ApSupplier] [AS]
Where [AS].[ActivePOFlag]=1
GO
EXEC sp_addextendedproperty N'MS_Description', N'limit of Ap suppliers for use in
look ups', 'SCHEMA', N'dbo', 'VIEW', N'vw_APSuppliersWithActivePO', NULL, NULL
GO
```

Uses

[Lookups].[ApSupplier]

Author: Johnson, Chris

[Lookups].[StockCodes_Bought]

Columns

Name
Company
StockCode
CompanyStockCode
StockDescription

SQL Script

```
CREATE View [Lookups].[StockCodes_Bought]
as
SELECT [SC].[Company]
     , [SC].[StockCode]
     , [CompanyStockCode] = [SC].[Company]+' - '+[SC].[StockCode]
     , [SC].[StockDescription]
FROM [Lookups].[StockCode] [SC]
Where [SC].[PartCategory]='B'


GO
```

Uses

[Lookups].[StockCode]
[Lookups]

Author: Johnson, Chris

[Lookups].[vw_AnalysisCategoryDistinct]

Columns

Name
GlAnalysisCategory

SQL Script

```
Create View [Lookups].[vw_AnalysisCategoryDistinct]
As
    Select Distinct
            [GAC].[GlAnalysisCategory]
    From    [Lookups].[GlAnalysisCategory] [GAC]
GO
```

Uses

[Lookups].[GlAnalysisCategory]
[Lookups]

Author: Johnson, Chris

[Lookups].[vw_APSuppliersWithActivePO]

MS_Description

limit of Ap suppliers for use in look ups

Columns

Name
Supplier
SupplierName

SQL Script

```
Create View [Lookups].[vw_APSuppliersWithActivePO]
As
Select Distinct [AS].[Supplier]
     , [AS].[SupplierName]
From [Lookups].[ApSupplier] [AS]
Where [AS].[ActivePOFlag]=1
GO
EXEC sp_addextendedproperty N'MS_Description', N'limit of Ap suppliers for use in
look ups', 'SCHEMA', N'Lookups', 'VIEW', N'vw_APSuppliersWithActivePO', NULL, NULL
GO
```

Uses

[Lookups].[ApSupplier]
[Lookups]

Author: Johnson, Chris

[Lookups].[vw_DistinctBuyerList]

MS_Description

limit of Buyers for use in look ups

Columns

Name
BuyerName

SQL Script

```
Create View [Lookups].[vw_DistinctBuyerList]
as
SELECT Distinct
[B].[BuyerName] FROM [Lookups].[Buyers] As [B]
GO
EXEC sp_addextendedproperty N'MS_Description', N'limit of Buyers for use in look
ups', 'SCHEMA', N'Lookups', 'VIEW', N'vw_DistinctBuyerList', NULL, NULL
GO
```

Uses

[Lookups].[Buyers]
[Lookups]

Author: Johnson, Chris

[Lookups].[vw_StockCodes_Bought_ActivePO]

MS_Description

limit of StockCodes for use in look ups

Columns

Name
Company
StockCode
CompanyStockCode
StockDescription

SQL Script

```
Create View [Lookups].[vw_StockCodes_Bought_ActivePO]
As
    Select  [SC].[Company]
          , [SC].[StockCode]
          , [CompanyStockCode] = [SC].[Company] + ' - ' + [SC].[StockCode]
          , [SC].[StockDescription]
    From    [Lookups].[StockCode] [SC]
    Where   [SC].[PartCategory] = 'B'
            And [SC].[ActivePOFlag] = 1;


GO
EXEC sp_addextendedproperty N'MS_Description', N'limit of StockCodes for use in look
ups', 'SCHEMA', N'Lookups', 'VIEW', N'vw_StockCodes_Bought_ActivePO', NULL, NULL
GO
```

Uses

[Lookups].[StockCode]
[Lookups]

Author: Johnson, Chris

Stored Procedures

Objects

Name
Process.ExecForEachDB The purpose of this stored procedure is to replace the undocumented procedure sp_MSforeachdb as this may be removed in future versions of SQL Server. The stored procedure iterates through all user databases and executes the code passed to it.
Process.ExecForEachDB_WithTableCheck The purpose of this stored procedure is to replace the undocumented procedure sp_MSforeachdb as this may be removed in future versions of SQL Server. The stored procedure iterates through all user databases and executes the code passed to it. In addition to this, proc checks that tables exist before executing (reducing error messages)
Process.UspAlter_AddColumn procedure to alter tables by adding columns, used in history table generation
Process.UspAlter_CheckAndAddColumns works out which columns need to be added to tables for history generation
Process.UspAlter_TableName used to rename tables and drops indexes (used for automatic archiving of history tables)
Process.UspCreate_Table used to create tables and constraints (used for automatic creation of history tables)
Process.UspCreateAmend_HistoryTables proc to automatically create and amend history tables in BlackBox
Process.UspInsert_RedTagLogs procedure is used to drive logs from reports into table RedTagLogs table
Process.UspLoad_LoadController Stored procedure to execute all stored procs marked as update
Process.UspPopulate_HistoryTables1 populates history tables from SysproTransactionsLogged
Process.UspPopulate_UnpivotHistory Stored proc to update specified tables, creating new fields on the fly
Process.UspUpdate_ApSupplier Stored proc to update specified table
Process.UspUpdate_ArInvoicePayTrnType Stored proc to update specified table
Process.UspUpdate_BankBalances Stored proc to update specified table
Process.UspUpdate_Bin Stored proc to update specified table
Process.UspUpdate_BudgetType Stored proc to update specified table
Process.UspUpdate_Buyers Stored proc to update specified table
Process.UspUpdate_CompanyNames Stored proc to update specified table
Process.UspUpdate_CurrencyRates Stored proc to update specified table
Process.UspUpdate_GenJournalCtlJnlSource Stored proc to update specified table
Process.UspUpdate_GenTransactionSource Stored proc to update specified table
Process.UspUpdate_GlAccountCode

Author: Johnson, Chris

Stored proc to update specified table
Process.UspUpdate_GLAccountType Stored proc to update specified table
Process.UspUpdate_GlAnalysisCategories Stored proc to update specified table
Process.UspUpdate_GlExpenseCode Stored proc to update specified table
Process.UspUpdate_InvMaster_PartCategory Stored proc to update specified table
Process.UspUpdate_MCompleteFlag Stored proc to update specified table
Process.UspUpdate_PorLineType Stored proc to update specified table
Process.UspUpdate_ProductClass Stored proc to update specified table
Process.UspUpdate_PurchaseOrderInvoiceMapping Stored proc to update specified table
Process.UspUpdate_PurchaseOrderStatus Stored proc to update specified table
Process.UspUpdate_PurchaseOrderTaxStatus Stored proc to update specified table
Process.UspUpdate_PurchaseOrderType Stored proc to update specified table
Process.UspUpdate_ReqBuyers Stored proc to update specified table
Process.UspUpdate_ReqnStatus Stored proc to update specified table
Process.UspUpdate_SalesOrderDocumentType Stored proc to update specified table
Process.UspUpdate_SalesOrderLineType Stored proc to update specified table
Process.UspUpdate_SalesOrderStatus Stored proc to update specified table
Process.UspUpdate_StockCode Stored proc to update specified table
Process.UspUpdate_TrnTypeModifier Stored proc to update specified table
Process.UspUpdate_Warehouse Stored proc to update specified table
Report.Results_AllCurrencyRates provides a list of currency rates for all time
Report.Results_BudgetTest unused procedure
Report.UspResults_AccPayable_AgedInvoices returns details of aged accounts payable invoices
Report.UspResults_AllBankBalances returns list of all bank balances
Report.UspResults_AllPosAndReqs list of all purchase orders and requisitions
Report.UspResults_AllSalesOrders list of all sales orders
Report.UspResults_AllSuppliers list of all suppliers
Report.UspResults_AmendmentJnl details from the amendment jnl

Report.UspResults_ApAgingInvoices replaced by UspResults_AccPayable_AgedInvoices
Report.UspResults_APDaysToPayment returns details on how long it takes for AP to be settled
Report.UspResults_ApGlDisburse details of the general ledger distribution from accounts payable
Report.UspResults_ApInvoice details of Ap Invoices
Report.UspResults_ApInvoicesWithPaymentDetails details of Ap Invoices including payment details
Report.UspResults_ApJnlGLDistrs details of the general ledger distribution from accounts payable with journals
Report.UspResults_ApSupplierNar used to return AP supp narrative in documents
Report.UspResults_ApSuppNarr used to return AP supp narrative in documents
Report.UspResults_ArCustomers list of all AR customers
Report.UspResults_ARDaysToPayment returns details on how long it takes for AR to be settled
Report.UspResults_AvailableLots list of all available lots
Report.UspResults_AvailableLots_AmountRequired list of available lots up to the required amount entered, lots picked from oldest to newest
Report.UspResults_AvailableStock list of stock available in each warehouse
Report.UspResults_BudgetsActuals budgets vs actuals from GL
Report.UspResults_BudgetsVsActuals budgets vs actuals from GL
Report.UspResults_ClosingBalancesInterco Return closing balances related to InterCompany payments for analysis
Report.UspResults_CompaniesCurrencyLatestExchangeRates returns current exchange rates in use
Report.UspResults_CompanyTransactions journals for each company
Report.UspResults_ExecForEachDBLogs returns details of how often exec for each db is being called, used to monitor code usage
Report.UspResults_FixedAssets details of all fixed assets
Report.UspResults_GeneralLedgerControlStats details from Gen Control tables, providing quick overview of ledger
Report.UspResults_GenJournalDetails list of journals
Report.UspResults_GenJournalEntries list of journal entries
Report.UspResults_GenledgerJournalsGrouped list of journals grouped
Report.UspResults_GenMasterMapping not in use
Report.UspResults_GL_Actuals list of gl actual figures
Report.UspResults_GL_Budgets list of gl budget figures
Report.UspResults_GL_Codes list of gl codes

Author: Johnson, Chris

Report.UspResults_GLMovements list of gl movements
Report.UspResults_GLMovementsIntercoPL Intercompany profit and loss general ledger
Report.UspResults_GLMovementsPBLRevCos PBL Revenue and Cost of Sales
Report.UspResults_GrnInvoiceDetails provides details of goods received in relation to invoices
Report.UspResults_InventoryInInspection list of inventory in inspection
Report.UspResults_InventoryInspectionTimes list of inventory in inspection with completion times
Report.UspResults_InvoiceRunVerify not in use
Report.UspResults_JobHeader high level details of Job
Report.UspResults_JobHeader_AllJobs high level details of Job, for all jobs
Report.UspResults_JobStockDetails list of stock related with job
Report.UspResults_JournalDetails summary journal info
Report.UspResults_Labour not in use
Report.UspResults_LabourDetails list of labour
Report.UspResults_LabourGlCodes not in use
Report.UspResults_LabourLoggedPerISOWeek labour logged split by ISO week
Report.UspResults_ListAllCompanies list of all companies available
Report.UspResults_LotsRetesting list of lots and when they need to be retested
Report.UspResults_LotTraceability searches for all lot transactions
Report.UspResults_LotTraceability_WithRuntimes searches for all lot transactions
Report.UspResults_MovementsTrialBalances not in use
Report.UspResults_OpenPurchaseOrderStock returns list of open purchase orders with stock details
Report.UspResults_OutstandingPurchaseOrders list of purchase orders outstanding
Report.UspResults_PaymentRunVerify data for Payment Run report
Report.UspResults_PosNoReqs details of Purchase orders without requisitions
Report.UspResults_PurchaseOrderChanges list of changes made to purchase orders
Report.UspResults_PurchaseOrderDetails purchase order details
Report.UspResults_PurchaseOrders purchase order details
Report.UspResults_PurchaseOrdersHistory list of changes made to purchase orders

Author: Johnson, Chris

Report.UspResults_PurchaseOrdersInvoices purchase order details with invoices
Report.UspResults_PurchaseOrdersOpen purchase order details for open purchase orders
Report.UspResults_PurchaseOrderWithHistory purchase order details with purchase order changes
Report.UspResults_ReceivedLotJob
Report.UspResults_RedTagDetails provides details of red tag logs (logs generated by other SPs)
Report.UspResults_ReportSysRefresh_RedTagLogs returns red tag logs for sys refresh times to ensure this is working and the average time a run takes
Report.UspResults_ReqsAfterInvoices list of requisitions raised post invoices appearing in system
Report.UspResults_RequisitionStatus list of requisitions with status
Report.UspResults_RequisitionUsers list of requisitions users
Report.UspResults_ReservedLots list of reserved lots used (allocated blood)
Report.UspResults_SalesOrderKPI data for sales order KPI report
Report.UspResults_SalesOrders list of sales orders
Report.UspResults_SalesOrderStats data for sales order stats report
Report.UspResults_ScrapCosts list of scrap costs
Report.UspResults_SearchForMissingIndexes proc to evaluate any missing indexes that can be added
Report.UspResults_StockDispatches details of stock dispatched
Report.UspResults_StockLevels stock levels and locations
Report.UspResults_StockMovements list of all stock movements
Report.UspResults_StockOutput list of job output stock
Report.UspResults_StockOutputDispatches not in use
Report.UspResults_StockOutputDispatches2 not in use
Report.UspResults_StockWithRunTimes list of stock, including the labour time to generate stock
Report.UspResults_Template used to develop new stored procs (copy and paste code and alter as necessary)
Report.UspResults_TrialBalance not in use
Report.UspResults_UnpaidAssets list of unpaid assets
Report.UspResults_UnpaidGRNAssets list of unpaid assets in GRN
Report.UspResults_WipStockRequiredPerJob list of stock required for each wip job
Report.UspResults_WipSubJobStock iterates through all jobs and subjobs, returning their stock - used in picklist/bill of materials reports

Author: Johnson, Chris

Report.UspResultsRefresh_TableTimes used to refresh all the lookup and history table, in addition providing times for how long this takes
Reports.UspResults_SearchForProcedures proc to search for stored procedures
Reports.UspResults_SearchForText proc to search for text in dbs - warning this is server intensive
Review.UspResults_CompanyDuplicates check for duplicates in companyname table
Review.UspResults_CompanyNamesMissing check for missing company names in lookup
Review.UspResults_CurrencyRatesMissing check that currency rates are available
Review.UspResults_DbDiffs provide details of differences between databases
Review.UspResults_LastUpdatedTimes check lookup and history tables for when last updated
Review.UspResults_LogStats review of red tag logs
Review.UspResults_LookupDuplicates check lookup tables for duplicates (indicates that refresh failed)
Review.UspResults_ProcsMostRun details of the most commonly used stored procedures
Review.UspResults_ReportsDevelopmentRun details of reports in development run
Review.UspResults_ReportsMostRun details of the most commonly used reports
Review.UspResults_SysproTransactionsCheck check of non-entered audit logs into history tables
Review.UspResults_SysproTransactionsTrending review of syspro audit logs captured in BlackBox
Review.UspResults_WarehousesIssueFrom details of warehouse that do not have issue from completed

[Process].[ExecForEachDB]

MS_Description

The purpose of this stored procedure is to replace the undocumented procedure sp_MSforeachdb as this may be removed in future versions
of SQL Server. The stored procedure iterates through all user databases and executes the code passed to it.

Parameters

Name	Data Type	Max Length (Bytes)
@cmd	nvarchar(max)	max

SQL Script

```
CREATE Proc [Process].[ExecForEachDB] ( @cmd NVarchar(Max) )
As /*
Stored Procedure created by Chris Johnson
20th January 2016

The purpose of this stored procedure is to replace the undocumented procedure sp_-
MSforeachdb as this may be removed in future versions
of SQL Server. The stored procedure iterates through all user databases and executes
the code passed to it.

Based off of http://sqlblog.com/blogs/aaron_bertrand/archive/2010/02/08/bad-habits-
to-kick-relying-on-undocumented-behavior.aspx
*/
    Begin
        Set NoCount On;

    --Declare variables

        Declare @SqlScript NVarchar(Max)= ''
          , @Database NVarchar(257)=''
          , @ErrorMessage NVarchar(Max)='';

    --Try to create Logging table (if permissions to create exist)

        If Not Exists ( Select  1
                        From    [sys].[tables] [T]
                                Left Join [sys].[schemas] As [S] On [S].[schema_id]
= [T].[schema_id]
                        Where   [S].[name] = 'History'
                                And [T].[name] = 'ExecForEachDBLogs' )
            Begin
                Begin Try
                    Create Table [History].[ExecForEachDBLogs]
                        (
                          [LogID] BigInt Identity(1 , 1)
                        , [LogTime] DateTime2 Default GetDate()
                        , [Cmd] NVarchar(2000) collate Latin1_General_BIN
                        );
                End Try
```

Author: Johnson, Chris

```
                Begin Catch
                    Print 'unable to create logging table';
                End Catch;
            End;

    --Add Logging details

        If Object_Id('[History].[ExecForEachDBLogs]') Is Not Null
            Begin
                Begin Try
                    Insert  [History].[ExecForEachDBLogs]
                            ( [Cmd] )
                    Values  ( @cmd );
                End Try
                Begin Catch
                    Print 'unable to capture logging details';
                End Catch;
            End;

    --Try to create error Logging table (if permissions to create exist)

        If Not Exists ( Select  1
                        From    [sys].[tables] [T]
                                Left Join [sys].[schemas] As [S] On [S].[schema_id]
= [T].[schema_id]
                        Where   [S].[name] = 'History'
                                And [T].[name] = 'ExecForEachDBErrorLogs' )
            Begin
                Begin Try
                    Create Table [History].[ExecForEachDBErrorLogs]
                        (
                          [LogID] BigInt Identity(1 , 1)
                        , [LogTime] DateTime2 Default GetDate()
                        , [Error] NVarchar(2000)collate Latin1_General_BIN
                        );
                End Try
                Begin Catch
                    Print 'unable to create error logging table';
                End Catch;
            End;


    --Test validity, all scripts should contain a "?" to be used in place of a db
name

        If @cmd Not Like '%?%'
            Begin
                Set @ErrorMessage = Cast('' As NVarchar(max))
                Set @ErrorMessage = @ErrorMessage+'ExecForEachDB failed, script does
not contain the string "?" '
                    + @cmd;

                --If is included as permissions may not be available to create this
table

                If Object_Id('[History].[ExecForEachDBLogs]') Is Not Null
                    Begin
                        Insert  [History].[ExecForEachDBErrorLogs]
```

Author: Johnson, Chris

```
                                ( [Error] )
                        Values  ( @ErrorMessage );
                    End;

                If Object_Id('[History].[ExecForEachDBLogs]') Is Null
                    Begin
                        Raiserror ('** Warning - Errors are not being logged
**',1,1); --if Errors are not being logged raise a low level error

                    End;
                Raiserror (@ErrorMessage,13,1);
            End;

        If @cmd Like '%?%'
            Begin
    --Use Cursor to hold list of databases to execute against

                Declare [DbNames] Cursor Local Forward_Only Static Read_Only
                For
                    Select  QuoteName([name])
                    From    [sys].[databases]
                    Where   [state] = 0 --online databases

                            And [is_read_only] = 0 --only databases that can be
executed against

                            And [database_id] > 4 --only user databases

                            And has_dbaccess([name]) = 1 --only dbs current user has
access to

                    Order By [name];

                Open [DbNames];

                Fetch Next From [DbNames] Into @Database; --Get first database to
execute against


                While @@fetch_status = 0 --when fetch is successful

                    Begin
                        Set @SqlScript = Cast('' As NVarchar(Max));
                        Set @SqlScript = @SqlScript
                            + Replace(Replace(Replace(@cmd , '?' , @Database) ,
                                        '[[' , '[') , ']]' , ']');--[[ & ]]
caused by script including [?]

                        Begin Try
                            Exec(@SqlScript);
                        End Try
                        Begin Catch --if error happens against any db, raise a high
level error advising the database and print the script

                            Set @ErrorMessage = Cast('' As NVarchar(max))
                            Set @ErrorMessage = @ErrorMessage + 'Script failed
against database '
                                + @Database;
                            Raiserror (@ErrorMessage,13,1);
```

Author: Johnson, Chris

```
                            Print @SqlScript;
                        End Catch;

                        Fetch Next From [DbNames] Into @Database;--Get next database
to execute against

                    End;

                Close [DbNames];
                Deallocate [DbNames];
            End;
    End;
GO
EXEC sp_addextendedproperty N'MS_Description', N'The purpose of this stored
procedure is to replace the undocumented procedure sp_MSforeachdb as this may be
removed in future versions
of SQL Server. The stored procedure iterates through all user databases and executes
the code passed to it.', 'SCHEMA', N'Process', 'PROCEDURE', N'ExecForEachDB', NULL,
NULL
GO
```

Uses

[History].[ExecForEachDBErrorLogs]
[History].[ExecForEachDBLogs]
[Process]

Used By

[Process].[UspUpdate_BankBalances]
[Process].[UspUpdate_Bin]
[Process].[UspUpdate_Buyers]
[Process].[UspUpdate_CompanyNames]
[Process].[UspUpdate_GlAnalysisCategories]
[Process].[UspUpdate_MCompleteFlag]
[Process].[UspUpdate_ProductClass]
[Process].[UspUpdate_PurchaseOrderInvoiceMapping]
[Process].[UspUpdate_PurchaseOrderStatus]
[Process].[UspUpdate_PurchaseOrderTaxStatus]
[Process].[UspUpdate_PurchaseOrderType]
[Process].[UspUpdate_ReqBuyers]
[Process].[UspUpdate_ReqnStatus]
[Process].[UspUpdate_SalesOrderLineType]
[Process].[UspUpdate_SalesOrderStatus]
[Process].[UspUpdate_TrnTypeModifier]
[Process].[UspUpdate_Warehouse]
[Report].[UspResults_AllBankBalances]
[Report].[UspResults_AllPosAndReqs]
[Report].[UspResults_AmendmentJnl]
[Report].[UspResults_ApAgingInvoices]
[Report].[UspResults_ApSupplierNar]
[Report].[UspResults_ArCustomers]
[Report].[UspResults_ARDaysToPayment]
[Report].[UspResults_AvailableLots]
[Report].[UspResults_AvailableLots_AmountRequired]
[Report].[UspResults_AvailableStock]
[Report].[UspResults_CompaniesCurrencyLatestExchangeRates]

Author: Johnson, Chris

[Report].[UspResults_CompanyTransactions]
[Report].[UspResults_GenJournalDetails]
[Report].[UspResults_GenJournalEntries]
[Report].[UspResults_GenledgerJournalsGrouped]
[Report].[UspResults_GrnInvoiceDetails]
[Report].[UspResults_InventoryInspectionTimes]
[Report].[UspResults_InvoiceRunVerify]
[Report].[UspResults_JobHeader]
[Report].[UspResults_JobHeader_AllJobs]
[Report].[UspResults_JobStockDetails]
[Report].[UspResults_Labour]
[Report].[UspResults_LabourDetails]
[Report].[UspResults_ListAllCompanies]
[Report].[UspResults_LotsRetesting]
[Report].[UspResults_OpenPurchaseOrderStock]
[Report].[UspResults_OutstandingPurchaseOrders]
[Report].[UspResults_PaymentRunVerify]
[Report].[UspResults_PurchaseOrderChanges]
[Report].[UspResults_PurchaseOrderDetails]
[Report].[UspResults_PurchaseOrders]
[Report].[UspResults_PurchaseOrdersHistory]
[Report].[UspResults_PurchaseOrdersInvoices]
[Report].[UspResults_PurchaseOrdersOpen]
[Report].[UspResults_PurchaseOrderWithHistory]
[Report].[UspResults_ReqsAfterInvoices]
[Report].[UspResults_RequisitionStatus]
[Report].[UspResults_RequisitionUsers]
[Report].[UspResults_ReservedLots]
[Report].[UspResults_SalesOrderKPI]
[Report].[UspResults_SalesOrders]
[Report].[UspResults_ScrapCosts]
[Report].[UspResults_StockDispatches]
[Report].[UspResults_StockLevels]
[Report].[UspResults_StockMovements]
[Report].[UspResults_StockOutput]
[Report].[UspResults_StockOutputDispatches2]
[Report].[UspResults_UnpaidAssets]
[Report].[UspResults_UnpaidGRNAssets]
[Report].[UspResults_WipStockRequiredPerJob]
[Reports].[UspResults_SearchForProcedures]
[Reports].[UspResults_SearchForText]
[Review].[UspResults_CompanyNamesMissing]
[Review].[UspResults_DbDiffs]
[Review].[UspResults_WarehousesIssueFrom]

[Process].[ExecForEachDB_WithTableCheck]

MS_Description

The purpose of this stored procedure is to replace the undocumented procedure sp_MSforeachdb as this may be removed in future versions
of SQL Server. The stored procedure iterates through all user databases and executes the code passed to it. In addition to this, proc checks that tables exist before executing (reducing error messages)

Parameters

Name	Data Type	Max Length (Bytes)
@cmd	nvarchar(max)	max
@SchemaTablesToCheck	nvarchar(max)	max

SQL Script

```
CREATE Proc [Process].[ExecForEachDB_WithTableCheck]
    (
      @cmd NVarchar(Max)
    , @SchemaTablesToCheck NVarchar(Max)
    )
As /*
Stored Procedure created by Chris Johnson
20th January 2016

The purpose of this stored procedure is to replace the undocumented procedure sp_-
MSforeachdb as this may be removed in future versions
of SQL Server. The stored procedure iterates through all user databases and executes
the code passed to it.

Based off of http://sqlblog.com/blogs/aaron_bertrand/archive/2010/02/08/bad-habits-
to-kick-relying-on-undocumented-behavior.aspx

Amended 21st June to check for tables to run against before executing
*/
    Begin
        Set NoCount On;

    --Declare variables

        Declare @SqlScript NVarchar(Max)= ''
          , @SqlValidationScript NVarchar(Max)= ''
          , @Database NVarchar(257)= ''
          , @ErrorMessage NVarchar(Max)= ''
          , @ValidationQuery NVarchar(Max)= 'declare @Temp int; Use [?]; '
          , @ValidQuery Int;

        Select  @ValidationQuery = @ValidationQuery
                + Stuff((Select 'SELECT @Temp=max(1)  FROM '
                                + QuoteName([Value]) + '
'
                         From   [dbo].[udf_SplitString](@SchemaTablesToCheck ,
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Process.ExecForEach-DB_WithTableCheck

```
                                                        ',')
                For     Xml Path('')
                          , Type
      ).[value]('text()[1]' , 'varchar(max)') , 1 , 0 , '');


       Select  @ValidQuery = 1;


    --Try to create Logging table (if permissions to create exist)

       If Not Exists ( Select  1
                       From    [sys].[tables] [T]
                               Left Join [sys].[schemas] As [S]
                                   On [S].[schema_id] = [T].[schema_id]
                       Where   [S].[name] = 'History'
                               And [T].[name] = 'ExecForEachDBLogs' )
           Begin
               Begin Try
                   Create Table [History].[ExecForEachDBLogs]
                       (
                         [LogID] BigInt Identity(1 , 1)
                       , [LogTime] DateTime2 Default GetDate()
                       , [Cmd] NVarchar(2000) Collate Latin1_General_BIN
                       );
               End Try
               Begin Catch
                   Print 'unable to create logging table';
               End Catch;
           End;

    --Add Logging details

       If Object_Id('[History].[ExecForEachDBLogs]') Is Not Null
           Begin
               Begin Try
                   Insert  [History].[ExecForEachDBLogs]
                           ( [Cmd] )
                   Values  ( @cmd );
               End Try
               Begin Catch
                   Print 'unable to capture logging details';
               End Catch;
           End;

    --Try to create error Logging table (if permissions to create exist)

       If Not Exists ( Select  1
                       From    [sys].[tables] [T]
                               Left Join [sys].[schemas] As [S]
                                   On [S].[schema_id] = [T].[schema_id]
                       Where   [S].[name] = 'History'
                               And [T].[name] = 'ExecForEachDBErrorLogs' )
           Begin
               Begin Try
                   Create Table [History].[ExecForEachDBErrorLogs]
                       (
```

Author: Johnson, Chris

```
                            [LogID] BigInt Identity(1 , 1)
                          , [LogTime] DateTime2 Default GetDate()
                          , [Error] NVarchar(2000) Collate Latin1_General_BIN
                          );
                End Try
                Begin Catch
                    Print 'unable to create error logging table';
                End Catch;
            End;


    --Test validity, all scripts should contain a "?" to be used in place of a db
name

        If @cmd Not Like '%?%'
            Begin
                Set @ErrorMessage = Cast('' As NVarchar(Max));
                Set @ErrorMessage = @ErrorMessage
                    + 'ExecForEachDB failed, script does not contain the string "?"
'
                    + @cmd;

                --If is included as permissions may not be available to create this
table

                If Object_Id('[History].[ExecForEachDBLogs]') Is Not Null
                    Begin
                        Insert  [History].[ExecForEachDBErrorLogs]
                                ( [Error] )
                        Values  ( @ErrorMessage );
                    End;

                If Object_Id('[History].[ExecForEachDBLogs]') Is Null
                    Begin
                        Raiserror ('** Warning - Errors are not being logged
**',1,1); --if Errors are not being logged raise a low level error

                    End;
                Raiserror (@ErrorMessage,13,1);
            End;

        If @cmd Like '%?%'
            Begin
    --Use Cursor to hold list of databases to execute against

                Declare [DbNames] Cursor Local Forward_Only Static Read_Only
                For
                    Select  QuoteName([name])
                    From    [sys].[databases]
                    Where   [state] = 0 --online databases

                            And [is_read_only] = 0 --only databases that can be
executed against

                            And [database_id] > 4 --only user databases

                            And Has_DbAccess([name]) = 1 --only dbs current user has
access to
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Process.ExecForEach-DB_WithTableCheck

```
                    Order By [name];

                Open [DbNames];

                Fetch Next From [DbNames] Into @Database; --Get first database to
execute against


                While @@fetch_status = 0 --when fetch is successful

                    Begin
                        Set @SqlScript = Cast('' As NVarchar(Max));
                        Set @SqlScript = @SqlScript
                            + Replace(Replace(Replace(@cmd , '?' , @Database) ,
                                              '[[' , '[') , ']]' , ']');--[[ & ]]
caused by script including [?]

                        Set @SqlValidationScript = Cast('' As NVarchar(Max));
                        Set @SqlValidationScript = @SqlValidationScript
                            + Replace(Replace(Replace(@ValidationQuery , '?' ,
                                                      @Database) , '[[' , '[') ,
                                      ']]' , ']');--[[ & ]] caused by script
including [?]


                        Set @ValidQuery = 1;
                        Begin Try

                            Exec (@SqlValidationScript);

                        End Try
                        Begin Catch
                            Set @ValidQuery = 0;
                        End Catch;
                        If @ValidQuery = 1
                            Begin Try

                                Exec(@SqlScript);

                            End Try
                            Begin Catch --if error happens against any db, raise a
high level error advising the database and print the script

                                Set @ErrorMessage = Cast('' As NVarchar(Max));
                                Set @ErrorMessage = @ErrorMessage
                                    + 'Script failed against database '
                                    + @Database;
                                Raiserror (@ErrorMessage,13,1);
                                Print @SqlScript;
                            End Catch;

                        Fetch Next From [DbNames] Into @Database;--Get next database
to execute against

                    End;

                Close [DbNames];
```

Author: Johnson, Chris

```
                Deallocate [DbNames];
            End;
    End;
GO
EXEC sp_addextendedproperty N'MS_Description', N'The purpose of this stored
procedure is to replace the undocumented procedure sp_MSforeachdb as this may be
removed in future versions
of SQL Server. The stored procedure iterates through all user databases and executes
the code passed to it. In addition to this, proc checks that tables exist before
executing (reducing error messages)', 'SCHEMA', N'Process', 'PROCEDURE', N'ExecFor-
EachDB_WithTableCheck', NULL, NULL
GO
```

Uses

[dbo].[udf_SplitString]
[History].[ExecForEachDBErrorLogs]
[History].[ExecForEachDBLogs]
[Process]

Used By

[Process].[UspCreateAmend_HistoryTables]
[Process].[UspPopulate_HistoryTables1]
[Process].[UspUpdate_ApSupplier]
[Process].[UspUpdate_StockCode]
[Report].[UspResults_AllSalesOrders]
[Report].[UspResults_AllSuppliers]
[Report].[UspResults_APDaysToPayment]
[Report].[UspResults_ApGlDisburse]
[Report].[UspResults_ApInvoice]
[Report].[UspResults_ApInvoicesWithPaymentDetails]
[Report].[UspResults_ApJnlGLDistrs]
[Report].[UspResults_ApSuppNarr]
[Report].[UspResults_FixedAssets]
[Report].[UspResults_InventoryInInspection]
[Report].[UspResults_LabourLoggedPerISOWeek]
[Report].[UspResults_LotTraceability]
[Report].[UspResults_LotTraceability_WithRuntimes]
[Report].[UspResults_PosNoReqs]
[Report].[UspResults_ReceivedLotJob]
[Report].[UspResults_SalesOrderStats]
[Report].[UspResults_StockWithRunTimes]
[Report].[UspResults_Template]

Author: Johnson, Chris

[Process].[UspAlter_AddColumn]

MS_Description

procedure to alter tables by adding columns, used in history table generation

Parameters

Name	Data Type	Max Length (Bytes)
@Schema	varchar(500)	500
@Table	varchar(500)	500
@Column	varchar(500)	500
@Type	varchar(500)	500

SQL Script

```
CREATE Proc [Process].[UspAlter_AddColumn]
(@Schema Varchar(500)
,@Table Varchar(500)
,@Column Varchar(500)
,@Type Varchar(500))
As
/*
Template designed by Chris Johnson, Prometic Group January 2016
Stored procedure set out to create a column on a table
*/
 Begin
    Declare @SQLColumn Varchar(max) = 'Alter Table '+@Schema+'.'+@Table+' Add
'+QuoteName(@Column)+' '+@Type

    Exec (@SQLColumn)
    Print 'Column '+@Column+' added to table '+@Schema+'.'+@Table
 End


GO
EXEC sp_addextendedproperty N'MS_Description', N'procedure to alter tables by adding
columns, used in history table generation', 'SCHEMA', N'Process', 'PROCEDURE', N'Usp-
Alter_AddColumn', NULL, NULL
GO
```

Uses

[Process]

Used By

[Process].[UspAlter_CheckAndAddColumns]

Author: Johnson, Chris

[Process].[UspAlter_CheckAndAddColumns]

MS_Description

works out which columns need to be added to tables for history generation

SQL Script

```
CREATE Proc [Process].[UspAlter_CheckAndAddColumns]
As /*Template designed by Chris Johnson, Prometic Group September 2015
Stored procedure set out to check how many columns are missings */
    Begin
--remove nocount on to speed up query

        Set NoCount On;

        Declare @TotalColumns Int
          , @CurrentColumn Int
          , @ColumnName Varchar(500)
          , @TableName Varchar(500)
          , @ColumnType Varchar(500);

        Create Table [#ColumnsToCreate]
            (
              [CID] Int Identity(1 , 1)
            , [TableName] Varchar(500) Collate Latin1_General_BIN
            , [ColumnName] Varchar(500) Collate Latin1_General_BIN
            , [VariableType] Char(1) Collate Latin1_General_BIN
            );

            --Get list of columns that don't exist

        Insert  [#ColumnsToCreate]
                ( [TableName]
                , [ColumnName]
                , [VariableType]
                )
                Select  [TableName] = [tl].[TableName]
                      , [ColumnName] = Replace(Upper([tl].[VariableDesc]) ,
                                               ' ' , '')
                      , [tl].[VariableType]
                From    ( Select Distinct
                                  [STL].[TableName]
                                , [STL].[VariableDesc]
                                , [STL].[VariableType]
                          From      [Process].[SysproTransactionsLogged] As [STL]
                          Where     [STL].[AlreadyEntered] = 0
                        ) [tl]
                        Left Join [sys].[tables] As [T]
                            On [T].[name] = [tl].[TableName]
                        Left Join [sys].[schemas] As [S]
                            On [S].[schema_id] = [T].[schema_id]
                        Left Join [sys].[columns] As [C]
                            On [C].[object_id] = [T].[object_id]
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Process.UspAlter_Check-AndAddColumns

```
                                And [C].[name] = [tl].[VariableDesc]
                Where   [S].[name] = 'History'
                        And [T].[name] Not Like 'Archive%'
                        And [C].[name] Is Null;

                --Add new fields

        Select  @TotalColumns = Coalesce(Max([CTC].[CID]) , 0)
        From    [#ColumnsToCreate] As [CTC];

        If @TotalColumns > 0
            Begin
                Print 'Columns to be added '
                    + Cast(@TotalColumns As Varchar(50));
            End;
        If @TotalColumns = 0
            Begin
                Print 'No columns to be added';
            End;

        Set @CurrentColumn = 1;

        --iterate through each column that is required

        While @CurrentColumn <= @TotalColumns
            Begin
                Select  @ColumnName = [CTC].[ColumnName]
                      , @TableName = [CTC].[TableName]
                      , @ColumnType = Case When [CTC].[VariableType] = 'A'
                                           Then 'Varchar(255) collate
latin1_general_bin'
                                           When [CTC].[VariableType] = 'N'
                                           Then 'Float'
                                           When [CTC].[VariableType] = 'D'
                                           Then 'Date'
                                      End
                From    [#ColumnsToCreate] As [CTC]
                Where   [CTC].[CID] = @CurrentColumn;

                Exec [Process].[UspAlter_AddColumn] @Schema = 'History' , --
varchar(500)

                    @Table = @TableName , @Column = @ColumnName , -- varchar(500)

                    @Type = @ColumnType;

                Print 'History.' + @TableName + ' added column ' + @ColumnName
                    + ' ' + @ColumnType;
                Set @CurrentColumn = @CurrentColumn + 1;
            End;

    End;
GO
EXEC sp_addextendedproperty N'MS_Description', N'works out which columns need to be
added to tables for history generation', 'SCHEMA', N'Process', 'PROCEDURE', N'Usp-
Alter_CheckAndAddColumns', NULL, NULL
GO
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Process.UspAlter_Check-AndAddColumns

Uses

[Process].[SysproTransactionsLogged]

[Process].[UspAlter_AddColumn]

[Process]

Used By

[Process].[UspPopulate_HistoryTables1]

Author: Johnson, Chris

[Process].[UspAlter_TableName]

MS_Description

used to rename tables and drops indexes (used for automatic archiving of history tables)

Parameters

Name	Data Type	Max Length (Bytes)
@SchemaName	varchar(500)	500
@TableName	varchar(500)	500
@NewTableName	varchar(500)	500

SQL Script

```
CREATE Proc [Process].[UspAlter_TableName]
    (
      @SchemaName Varchar(500)
    , @TableName Varchar(500)
    , @NewTableName Varchar(500)
    )
As /*Template designed by Chris Johnson, Prometic Group January 2016
Stored procedure to rename tables dynamically
WARNING - THIS ALSO REMOVES Primary Keys constraints ASSOCIATED WITH TABLES RENAMED
*/
    Begin
        Set NoCount On;
        --before tables are moved, remove the indexes


        --get list of indexes attached to table

        Create Table [#Indexes]
            (
              [IID] Int Identity(1 , 1)
            , [ConstraintName] Varchar(500) collate latin1_general_bin
            );

        Insert  [#Indexes]
                ( [ConstraintName]
                )
                Select  [C].[name]
                From    [sys].[key_constraints] As [C]
                        Left Join [sys].[tables] As [T]
                            On [T].[object_id] = [C].[parent_object_id]
                        Left Join [sys].[schemas] As [S]
                            On [S].[schema_id] = [T].[schema_id]
                Where   [S].[name] = @SchemaName
                        And [T].[name] = @TableName
                        And [C].[name] Is Not Null;
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Process.UspAlter_Table-Name

```
        --get count of all indexes to be dropped, then loop through and remove them

        Declare @CurrentConstraint Int= 1
          , @CurrentConstraintName Varchar(500)
          , @TotalConstraints Int
          , @SQLConstraints Varchar(Max);

        Select  @TotalConstraints = Coalesce(Max([I].[IID]) , 0)
        From    [#Indexes] As [I];

        --Show how many indexes need to be removed

        Print @SchemaName + '.' + @TableName + ' Total Constraint to remove '
            + Cast(@TotalConstraints As Varchar(50));

        If @TotalConstraints > 0 --only iterate if there are indexes to remove

            Begin
                While @CurrentConstraint <= @TotalConstraints
                    Begin
                        Select  @CurrentConstraintName = [I].[ConstraintName]
                        From    [#Indexes] As [I]
                        Where   [I].[IID] = @CurrentConstraint;

                        Select  @SQLConstraints = 'Alter Table ' + @SchemaName
                                + '.' + @TableName + ' Drop CONSTRAINT '
                                + @CurrentConstraintName; --+ ' on '

                        --+ @SchemaName + '.' + @TableName;


                --Print @SQLConstraints;

                        Exec (@SQLConstraints);
                        Print 'Constraint ' + @CurrentConstraintName
                            + ' On table ' + @SchemaName + '.' + @TableName
                            + ' dropped.';
                        Set @CurrentConstraint = @CurrentConstraint + 1;
                    End;
            End;
        If @TotalConstraints = 0
            Begin
                Print 'No Constraints to drop On table ' + @SchemaName + '.'
                    + @TableName;
            End;


        Declare @ObjectName Varchar(1000)= @SchemaName + '.' + @TableName;
        --Once all indexes removed rename table

        Exec [sys].[sp_rename] @objname = @ObjectName ,
            @newname = @NewTableName , @objtype = 'OBJECT';

        Print @SchemaName + '.' + @TableName + ' renamed to ' + @NewTableName;
    End;
GO
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Process.UspAlter_Table-Name

```
EXEC sp_addextendedproperty N'MS_Description', N'used to rename tables and drops
indexes (used for automatic archiving of history tables)', 'SCHEMA', N'Process',
'PROCEDURE', N'UspAlter_TableName', NULL, NULL
GO
```

Uses

[Process]

Used By

[Process].[UspPopulate_HistoryTables1]

Author: Johnson, Chris

[Process].[UspCreate_Table]

MS_Description

used to create tables and constraints (used for automatic creation of history tables)

Parameters

Name	Data Type	Max Length (Bytes)
@SchemaName	varchar(500)	500
@TableName	varchar(500)	500
@Columns	varchar(500)	500
@Constraints	varchar(500)	500

SQL Script

```
CREATE Proc [Process].[UspCreate_Table]
    (
      @SchemaName Varchar(500)
    , @TableName Varchar(500)
    , @Columns Varchar(500)
    , @Constraints Varchar(500)
    )
As /*Template designed by Chris Johnson, Prometic Group January 2016
Stored procedure set out to create tables*/
    Begin
        Declare @SQLTable Varchar(Max)= 'Create TABLE ' + @SchemaName + '.'
            + @TableName + '(' + @Columns + '
     ,' + @Constraints + ')';

        Exec (@SQLTable);
        Print 'Table ' + @SchemaName + '.' + @TableName + ' created';
    End;
GO
EXEC sp_addextendedproperty N'MS_Description', N'used to create tables and
constraints (used for automatic creation of history tables)', 'SCHEMA', N'Process',
'PROCEDURE', N'UspCreate_Table', NULL, NULL
GO
```

Uses

[Process]

Used By

[Process].[UspPopulate_HistoryTables1]

Author: Johnson, Chris

[Process].[UspCreateAmend_HistoryTables]

MS_Description

proc to automatically create and amend history tables in BlackBox

Parameters

Name	Data Type	Max Length (Bytes)
@Rebuild	bit	1

SQL Script

```
CREATE Proc [Process].[UspCreateAmend_HistoryTables] ( @Rebuild Bit )
-- if rebuild = 1 then drop and recreate

-- if rebuild = 0 then

As /*
Template designed by Chris Johnson, Prometic Group September 2015
Stored procedure iterates through all electronic signatures and use the variable
description to unpivot the data
*/


    Set NoCount On;

--create temporary tables

    Create Table [#TableList]
        (
          [Tid] Int Identity(1 , 1)
        , [TableName] Varchar(150)    collate latin1_general_bin
        , Constraint [tlname] Primary Key NonClustered ( [TableName] )
            With ( Ignore_Dup_Key = On )
        );

    Create Table [#ColumnListTemp]
        (
          [Cid] Int Identity(1 , 1)
        , [ColumnName] Varchar(150)        collate latin1_general_bin
        , [ColumnType] Varchar(100)        collate latin1_general_bin
        , [TableName] Varchar(150)        collate latin1_general_bin
        , Constraint [cltdetailstemp] Primary Key NonClustered
            ( [ColumnName] , [ColumnType] , [TableName] )
            With ( Ignore_Dup_Key = On )
        );

    Create Table [#ColumnList]
        (
          [Cid] Int Identity(1 , 1)
        , [ColumnName] Varchar(150)        collate latin1_general_bin
```

Author: Johnson, Chris

```
        , [ColumnType] Varchar(100)         collate latin1_general_bin
        , [TableName] Varchar(150)         collate latin1_general_bin
        , Constraint [cldetails] Primary Key NonClustered
            ( [ColumnName] , [ColumnType] , [TableName] )
            With ( Ignore_Dup_Key = On )
        );

    Create Table [#CurrentColumns]
        (
          [ColumnName] Varchar(150)    collate latin1_general_bin
        , [TableName] Varchar(150)    collate latin1_general_bin
        );

    Insert  [#CurrentColumns]
            ( [ColumnName]
            , [TableName]
            )
            Select  [c].[name]
                  , [t].[name]
            From    [sys].[columns] [c]
                    Inner Join [sys].[tables] [t] On [t].[object_id] =
[c].[object_id]
                    Inner Join [sys].[schemas] [s] On [s].[schema_id] =
[t].[schema_id]
                                                And [s].[name] = 'History';

    Declare @ListOfTables Varchar(Max) = 'AdmSignatureLogDet,AdmSignatureLog';

--Get list of all tables that are monitored in the signature logs

    Declare @SQLTable Varchar(Max) = 'USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end'
        + --Only query DBs beginning SysProCompany

        '
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
            BEGIN
                        Insert  #TableList (TableName)
                        Select Distinct ASL.TableName
                        From dbo.AdmSignatureLog ASL
                        Left Join dbo.AdmSignatureLogDet ADSL
                            On ADSL.TransactionId = ASL.TransactionId
                               And ADSL.SignatureDate = ASL.SignatureDate
                               And ADSL.SignatureTime = ASL.SignatureTime
                               And ADSL.SignatureLine = ASL.SignatureLine
                               And ADSL.Operator = ASL.Operator
                        Left Join BlackBox.Lookups.AdmTransactionIDs ATI
                            On ATI.TransactionId = ASL.TransactionId
                        Where ASL.TableName <> ''''
                            And ADSL.VariableDesc <> ''''
                            And ( Case When ADSL.VarDateValue Is Null
                                            And ADSL.VarAlphaValue = '' ''
                                            And ADSL.VarNumericValue = 0 Then 1
                                       Else 0
                                  End ) = 0
            END
```

Author: Johnson, Chris

```
            ';

--Get list of all columns to be generated

    Declare @SQLColumn Varchar(Max) = 'USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end'
        + --Only query DBs beginning SysProCompany

        '
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
            BEGIN
            Insert  #ColumnListTemp
                    ( ColumnName
                    , ColumnType
                    , TableName
                    )
                    Select Distinct
                        replace(Upper(ADSL.VariableDesc),'' '','''')
                        , MAX(Case When ADSL.VariableType = ''A''
                                Then ''Varchar(255)''
                                When ADSL.VariableType = ''N''
                                Then ''Float''
                                When ADSL.VariableType = ''D''
                                Then ''Datetime''
                                Else ''Varchar(255)''
                            End)
                        , ASL.TableName
                    From
                        dbo.AdmSignatureLog ASL
                    Left Join dbo.AdmSignatureLogDet ADSL
                        On ADSL.TransactionId = ASL.TransactionId
                            And ADSL.SignatureDate = ASL.SignatureDate
                            And ADSL.SignatureTime = ASL.SignatureTime
                            And ADSL.SignatureLine = ASL.SignatureLine
                            And ADSL.Operator = ASL.Operator
                    Left Join BlackBox.Lookups.AdmTransactionIDs ATI
                        On ATI.TransactionId = ASL.TransactionId
                    Where
                        ADSL.VariableDesc <> ''''
                    Group By
                        ADSL.VariableDesc
                        , ASL.TableName
            END';

    Exec [Process].[ExecForEachDB_WithTableCheck] @cmd = @SQLTable , --
nvarchar(max)

        @SchemaTablesToCheck = @ListOfTables;
    Exec [Process].[ExecForEachDB_WithTableCheck]
     @cmd = @SQLColumn, -- nvarchar(max)

        @SchemaTablesToCheck = @ListOfTables;

    Update  [#ColumnListTemp]
    Set     [ColumnName] = Upper([ColumnName]);
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Process.UspCreate-Amend_HistoryTables

```
    Declare @Tablecount Int
      , @CurrentTable Int = 1
      , @TableName Varchar(150)
      , @ColumnCount Int
      , @CurrentColumn Int = 1
      , @SQL Varchar(Max)
      , @ColumnName Varchar(150)
      , @ColumnType Varchar(150);

--Number of tables to iterate through

    Select  @Tablecount = Max([Tid])
    From    [#TableList];

--Run through each table

    While @CurrentTable <= @Tablecount
        Begin
            Select  @TableName = [TableName]
            From    [#TableList]
            Where   [Tid] = @CurrentTable;

--drop the existing tables and recreate them

            If @Rebuild = 1
                Begin
                    Select  @SQL = 'If Exists(Select P.name FROM sys.tables P
                                    Left Join sys.schemas s On s.schema_id =
P.schema_id
                                    Where P.name=''' + @TableName
                            + ''' And s.name=''History'')
                        Begin
                            Drop table [History].' + @TableName + '
                        end

                        Create Table [History].[' + @TableName + ']
                        (' + Left(@TableName , 1) + 'ID int identity(1,1)
                        ,TransactionDescription varchar(150)
                        ,SignatureDatetime datetime2
                        ,Operator varchar(20)
                        ,ProgramName varchar(20)
                        ,Ranking bigint
                        ,ItemKey varchar(150)
                        ,DatabaseName varchar(150)
                        ,CONSTRAINT [' + @TableName
                            + '_ID] PRIMARY KEY (SignatureDatetime,Operator,Program-
Name,ItemKey,DatabaseName)
                            WITH (IGNORE_DUP_KEY = ON))'; --ignore duplication as
multiple rows to a change


                    Exec (@SQL);

                    --get

                    Insert  [#ColumnList]
                            ( [ColumnName]
```

Author: Johnson, Chris

```
                            , [ColumnType]
                            , [TableName]
                            )
                            Select Distinct
                                    [CT].[ColumnName]
                                  , [CT].[ColumnType]
                                  , @TableName
                            From    [#ColumnListTemp] [CT]
                            Where   [CT].[TableName] = @TableName;


                    Select  @ColumnCount = Max([Cid])
                    From    [#ColumnList];

                    Set @CurrentColumn = 1;
                    --SELECT * FROM #ColumnList


                    While @CurrentColumn <= @ColumnCount
                        Begin
                            Select  @ColumnName = [ColumnName]
                                  , @ColumnType = [ColumnType]
                            From    [#ColumnList]
                            Where   [Cid] = @CurrentColumn;

                            Select  @SQL = 'alter table [History].['
                                    + @TableName + '] add ['
                                    + Upper(@ColumnName) + '] ' + @ColumnType
                                    + ';';

                            Exec (@SQL);
                            Set @CurrentColumn = @CurrentColumn + 1;
                            Set @ColumnName = '';
                        End;

                    Set @CurrentTable = @CurrentTable + 1;
                    Truncate Table [#ColumnList];
                    Set @SQL = '';
                End;
            If @Rebuild = 0
                Begin
            --only create tables that have not previously existed

                    Select  @SQL = 'If Not Exists(Select P.name FROM sys.tables P
                                    Left Join sys.schemas s On s.schema_id =
P.schema_id
                                    Where P.name=''' + @TableName
                            + ''' And s.name=''History'')
                    Begin
                        Create Table [History].[' + @TableName + ']
                        (' + Left(@TableName , 1) + 'ID int identity(1,1)
                        ,TransactionDescription varchar(150) collate
latin1_general_bin
                        ,SignatureDatetime datetime2
                        ,Operator varchar(20) collate latin1_general_bin
                        ,ProgramName varchar(20) collate latin1_general_bin
                        ,Ranking bigint
```

Author: Johnson, Chris

```
                        ,ItemKey varchar(150) collate latin1_general_bin
                        ,CONSTRAINT [' + @TableName
                            + '_ID] PRIMARY KEY (SignatureDatetime,Operator,Program-
Name,ItemKey)
                            WITH (IGNORE_DUP_KEY = ON))
                        Create NonClustered Index IX_History_' + @TableName
                            + '_1 On [History].[' + @TableName
                            + '] (SignatureDatetime,Operator,ProgramName,ItemKey)
                    end'; --ignore duplication as multiple rows to a change


                    --Print @SQL;

                    Exec (@SQL);

                    --select only columns that do not exist

                    Insert  [#ColumnList]
                            ( [ColumnName]
                            , [ColumnType]
                            , [TableName]
                            )
                            Select Distinct
                                    [CT].[ColumnName]
                                  , [CT].[ColumnType]
                                  , @TableName
                            From    [#ColumnListTemp] [CT]
                                    Left Join [#CurrentColumns] [C] On [C].[Column-
Name] = [CT].[ColumnName]
                                                                       And [C].[TableName] =
[CT].[TableName]
                            Where   [CT].[TableName] = @TableName
                                    And [C].[ColumnName] Is Null;


                    Select  @ColumnCount = Max([Cid])
                    From    [#ColumnList];

                    Set @CurrentColumn = 1;

                    While @CurrentColumn <= @ColumnCount
                        Begin
                            Select  @ColumnName = [ColumnName]
                                  , @ColumnType = [ColumnType]
                            From    [#ColumnList]
                            Where   [Cid] = @CurrentColumn;

                            Select  @SQL = 'alter table [History].['
                                    + @TableName + '] add [' + @ColumnName
                                    + '] ' + @ColumnType + ';';

                            Exec (@SQL);
                            Set @CurrentColumn = @CurrentColumn + 1;
                            Set @ColumnName = '';
                        End;

                    Set @CurrentTable = @CurrentTable + 1;
```

Author: Johnson, Chris

```
                    --remove values from #ColumnList for next table

                    Truncate Table [#ColumnList];
                    Set @SQL = '';
                End;
        End;
GO
EXEC sp_addextendedproperty N'MS_Description', N'proc to automatically create and amend history tables in BlackBox', 'SCHEMA', N'Process', 'PROCEDURE', N'UspCreate-Amend_HistoryTables', NULL, NULL
GO
```

Uses

[Process].[ExecForEachDB_WithTableCheck]
[Process]

Author: Johnson, Chris

[Process].[UspInsert_RedTagLogs]

MS_Description

procedure is used to drive logs from reports into table RedTagLogs table

Parameters

Name	Data Type	Max Length (Bytes)
@StoredProcDb	varchar(255)	255
@StoredProcSchema	varchar(255)	255
@StoredProcName	varchar(255)	255
@UsedByType	char	1
@UsedByName	varchar(500)	500
@UsedByDb	varchar(255)	255

SQL Script

```
CREATE Proc [Process].[UspInsert_RedTagLogs]
    (
      @StoredProcDb Varchar(255)
    , @StoredProcSchema Varchar(255)
    , @StoredProcName Varchar(255)
    , @UsedByType Char(1)
    , @UsedByName Varchar(500)
    , @UsedByDb Varchar(255)
    )
As /*
Created by Chris Johnson 2nd February 2016
Stored proc to insert logs
*/
    Begin
        --create schemas if needed

        If Not Exists ( Select  1
                        From    [sys].[schemas] As [S]
                        Where   [S].[name] = 'Reports' )
            Begin
                Declare @Schema1 Varchar(500)= 'Create Schema Reports';
                Exec (@Schema1);
            End;
        If Not Exists ( Select  1
                        From    [sys].[schemas] As [S]
                        Where   [S].[name] = 'History' )
            Begin
                Declare @Schema2 Varchar(500)= 'Create Schema History';
                Exec (@Schema2);
            End;
        If Not Exists ( Select  1
                        From    [sys].[schemas] As [S]
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Process.UspInsert_Red-TagLogs

```
                        Where   [S].[name] = 'Lookups' )
            Begin
                Declare @Schema3 Varchar(500)= 'Create Schema Lookups';
                Exec (@Schema2);
            End;
        --create tables to capture logs

        If Not Exists ( Select  1
                        From    [sys].[tables] As [T]
                                Left Join [sys].[schemas] As [S]
                                    On [S].[schema_id] = [T].[schema_id]
                        Where   [T].[name] = 'RedTagsUsedByType'
                                And [S].[name] = 'Lookups' )
            Begin
                Create Table [Lookups].[RedTagsUsedByType]
                    (
                      [UsedByType] [Char](1) Not Null
                    , [UsedByDescription] [Varchar](150) collate latin1_general_bin
null
                    , Primary Key Clustered ( [UsedByType] Asc )
                        With ( Pad_Index = Off , Statistics_Norecompute = Off ,
                               Ignore_Dup_Key = Off , Allow_Row_Locks = On ,
                               Allow_Page_Locks = On ) On [PRIMARY]
                    )
                On  [PRIMARY];
                Insert  Into [Lookups].[RedTagsUsedByType]
                        ( [UsedByType]
                        , [UsedByDescription]
                        )
                Values  ( 'M'
                        , 'Manual Run'
                        );
                Insert  Into [Lookups].[RedTagsUsedByType]
                        ( [UsedByType]
                        , [UsedByDescription]
                        )
                Values  ( 'R'
                        , 'Report'
                        );
                Insert  Into [Lookups].[RedTagsUsedByType]
                        ( [UsedByType]
                        , [UsedByDescription]
                        )
                Values  ( 'S'
                        , 'Scheduled Run'
                        );

            End;
        If Not Exists ( Select  1
                        From    [sys].[tables] As [T]
                                Left Join [sys].[schemas] As [S]
                                    On [S].[schema_id] = [T].[schema_id]
                        Where   [T].[name] = 'RedTagLogs'
                                And [S].[name] = 'History' )
            Begin
                Create Table [History].[RedTagLogs]
                    (
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Process.UspInsert_Red-TagLogs

```
                        [TagID] [Int] Identity(1 , 1)
                                      Not Null
                    , [TagDatetime] [DateTime2](7) Null
                                                   Default ( GetDate() )
                    , [StoredProcDb] [Varchar](255)     collate latin1_general_bin
null
                    , [StoredProcSchema] [Varchar](255) collate latin1_general_bin
null
                    , [StoredProcName] [Varchar](255)    collate latin1_general_bin
null
                    , [UsedByType] [Char](1)            collate latin1_general_bin
                        Null
                        Foreign Key References [Lookups].[RedTagsUsedByType] ( [Used-
ByType] )
                    , [UsedByName] [Varchar](500)        collate latin1_general_bin
Null
                    , [UsedByDb] [Varchar](255)            collate
latin1_general_bin Null
                    )
                On  [PRIMARY];
            End;

        --Insert logs

        If Exists ( Select  1
                    From    [sys].[tables] As [T]
                            Left Join [sys].[schemas] As [S]
                                On [S].[schema_id] = [T].[schema_id]
                    Where   [T].[name] = 'RedTagLogs'
                            And [S].[name] = 'History' )
            Begin
                Begin Try
                    Insert  [History].[RedTagLogs]
                            ( [StoredProcDb]
                            , [StoredProcSchema]
                            , [StoredProcName]
                            , [UsedByType]
                            , [UsedByName]
                            , [UsedByDb]
                            )
                            Select  @StoredProcDb
                                  , @StoredProcSchema
                                  , @StoredProcName
                                  , Case When @UsedByType = '' Then 'X'
                                         Else @UsedByType
                                    End
                                  , @UsedByName
                                  , @UsedByDb;
                End Try
                Begin Catch
                    Raiserror('Red Tag error - failed to insert log',16,4);
                End Catch;
            End;
        --Raise Error if table doesn't exist

        If Not Exists ( Select  1
                        From    [sys].[tables] As [T]
                                Left Join [sys].[schemas] As [S]
```

Author: Johnson, Chris

```
                                    On [S].[schema_id] = [T].[schema_id]
                        Where    [T].[name] = 'RedTagLogs'
                                 And [S].[name] = 'History' )
            Begin
                Raiserror ('Red tag logs failure - tables do not exist',16,4);
            End;
    End;
GO
EXEC sp_addextendedproperty N'MS_Description', N'procedure is used to drive logs
from reports into table RedTagLogs table', 'SCHEMA', N'Process', 'PROCEDURE', N'Usp-
Insert_RedTagLogs', NULL, NULL
GO
```

Uses

[History].[RedTagLogs]
[Lookups].[RedTagsUsedByType]
[Process]

Used By

[Report].[Results_AllCurrencyRates]
[Report].[UspResults_AccPayable_AgedInvoices]
[Report].[UspResults_AllPosAndReqs]
[Report].[UspResults_AllSalesOrders]
[Report].[UspResults_AllSuppliers]
[Report].[UspResults_AmendmentJnl]
[Report].[UspResults_ApAgingInvoices]
[Report].[UspResults_APDaysToPayment]
[Report].[UspResults_ApGlDisburse]
[Report].[UspResults_ApInvoice]
[Report].[UspResults_ApInvoicesWithPaymentDetails]
[Report].[UspResults_ApJnlGLDistrs]
[Report].[UspResults_ApSuppNarr]
[Report].[UspResults_ArCustomers]
[Report].[UspResults_ARDaysToPayment]
[Report].[UspResults_AvailableLots]
[Report].[UspResults_AvailableLots_AmountRequired]
[Report].[UspResults_AvailableStock]
[Report].[UspResults_BudgetsActuals]
[Report].[UspResults_ClosingBalancesInterco]
[Report].[UspResults_CompaniesCurrencyLatestExchangeRates]
[Report].[UspResults_CompanyTransactions]
[Report].[UspResults_FixedAssets]
[Report].[UspResults_GeneralLedgerControlStats]
[Report].[UspResults_GenJournalDetails]
[Report].[UspResults_GenJournalEntries]
[Report].[UspResults_GenledgerJournalsGrouped]
[Report].[UspResults_GenMasterMapping]
[Report].[UspResults_GL_Codes]
[Report].[UspResults_GLMovements]
[Report].[UspResults_GLMovementsIntercoPL]
[Report].[UspResults_GLMovementsPBLRevCos]
[Report].[UspResults_GrnInvoiceDetails]
[Report].[UspResults_InventoryInInspection]
[Report].[UspResults_InventoryInspectionTimes]

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Process.UspInsert_Red-TagLogs

[Report].[UspResults_InvoiceRunVerify]
[Report].[UspResults_JobHeader]
[Report].[UspResults_JobHeader_AllJobs]
[Report].[UspResults_JobStockDetails]
[Report].[UspResults_JournalDetails]
[Report].[UspResults_Labour]
[Report].[UspResults_LabourDetails]
[Report].[UspResults_LabourGlCodes]
[Report].[UspResults_LabourLoggedPerISOWeek]
[Report].[UspResults_LotsRetesting]
[Report].[UspResults_LotTraceability]
[Report].[UspResults_LotTraceability_WithRuntimes]
[Report].[UspResults_MovementsTrialBalances]
[Report].[UspResults_OpenPurchaseOrderStock]
[Report].[UspResults_OutstandingPurchaseOrders]
[Report].[UspResults_PaymentRunVerify]
[Report].[UspResults_PosNoReqs]
[Report].[UspResults_PurchaseOrderChanges]
[Report].[UspResults_PurchaseOrderDetails]
[Report].[UspResults_PurchaseOrders]
[Report].[UspResults_PurchaseOrdersHistory]
[Report].[UspResults_PurchaseOrdersInvoices]
[Report].[UspResults_PurchaseOrdersOpen]
[Report].[UspResults_PurchaseOrderWithHistory]
[Report].[UspResults_ReceivedLotJob]
[Report].[UspResults_ReportSysRefresh_RedTagLogs]
[Report].[UspResults_ReqsAfterInvoices]
[Report].[UspResults_RequisitionStatus]
[Report].[UspResults_RequisitionUsers]
[Report].[UspResults_ReservedLots]
[Report].[UspResults_SalesOrderKPI]
[Report].[UspResults_SalesOrders]
[Report].[UspResults_SalesOrderStats]
[Report].[UspResults_ScrapCosts]
[Report].[UspResults_StockDispatches]
[Report].[UspResults_StockLevels]
[Report].[UspResults_StockMovements]
[Report].[UspResults_StockOutput]
[Report].[UspResults_StockOutputDispatches]
[Report].[UspResults_StockOutputDispatches2]
[Report].[UspResults_StockWithRunTimes]
[Report].[UspResults_Template]
[Report].[UspResults_TrialBalance]
[Report].[UspResults_UnpaidAssets]
[Report].[UspResults_UnpaidGRNAssets]
[Report].[UspResults_WipStockRequiredPerJob]
[Report].[UspResults_WipSubJobStock]
[Report].[UspResultsRefresh_TableTimes]
[Review].[UspResults_CompanyDuplicates]
[Review].[UspResults_CompanyNamesMissing]
[Review].[UspResults_CurrencyRatesMissing]
[Review].[UspResults_DbDiffs]
[Review].[UspResults_LastUpdatedTimes]
[Review].[UspResults_LogStats]
[Review].[UspResults_LookupDuplicates]
[Review].[UspResults_ProcsMostRun]
[Review].[UspResults_ReportsDevelopmentRun]

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Process.UspInsert_Red-TagLogs

[Review].[UspResults_ReportsMostRun]
[Review].[UspResults_SysproTransactionsCheck]
[Review].[UspResults_SysproTransactionsTrending]
[Review].[UspResults_WarehousesIssueFrom]
[Report].[UspResults_GLMovementsCenter]
[Report].[UspResults_MissingGroupMaps]
[Report].[UspResults_MissingRI2Groups]
[Report].[UspResults_MissingRI2Maps]

Author: Johnson, Chris

[Process].[UspLoad_LoadController]

MS_Description

Stored procedure to execute all stored procs marked as update

Parameters

Name	Data Type	Max Length (Bytes)
@HoursBetweenEachRun	numeric(5,2)	5

SQL Script

```
CREATE Proc [Process].[UspLoad_LoadController]
    (
      @HoursBetweenEachRun Numeric(5,2)
    )
As
    Begin
/*
Stored procedure created by Chris Johnson, Prometic Group September 2015 to execute
all update stored procs
*/
        Set NoCount On;
--find all procedures that need to be updated

        Create Table [#ProcsToRun]
            (
              [PID] Int Identity(1 , 1)
            , [SchemaName] Varchar(150)    collate Latin1_General_BIN
            , [ProcName] Varchar(150)    collate Latin1_General_BIN
            );

        Insert  [#ProcsToRun]
                ( [SchemaName]
                , [ProcName]
                )
                Select  [s].[name]
                      , [p].[name]
                From    [sys].[procedures] [p]
                        Left Join [sys].[schemas] [s]
                            On [s].[schema_id] = [p].[schema_id]
                Where   [s].[name] = 'Process'
                        And [p].[name] Like 'UspUpdate%';

        Declare @MaxProcs Int
          , @CurrentProc Int = 1;

        Select  @MaxProcs = Max([PID])
        From    [#ProcsToRun];

        Declare @SQL Varchar(Max)
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Process.UspLoad_Load-Controller

```
          , @SchemaName Varchar(150)
          , @ProcName Varchar(150);

    --run through each procedure, not caring if the count changes and only updating
if there have been more than # hours since the last run

        While @CurrentProc <= @MaxProcs
            Begin
                Select  @SchemaName = [SchemaName]
                      , @ProcName = [ProcName]
                From    [#ProcsToRun]
                Where   [PID] = @CurrentProc;

                Select  @SQL = 'begin try
begin tran
exec ' + @SchemaName + '.' + @ProcName
                        + ' @PrevCheck = 0,@HoursBetweenUpdates = '
                        + Cast(@HoursBetweenEachRun As Varchar(5)) + '
commit
end try
begin catch
rollback
end catch';
            --Print @SQL

                Exec (@SQL);
                Set @CurrentProc = @CurrentProc + 1;
            End;
    End;
GO
EXEC sp_addextendedproperty N'MS_Description', N'Stored procedure to execute all
stored procs marked as update ', 'SCHEMA', N'Process', 'PROCEDURE', N'UspLoad_Load-
Controller', NULL, NULL
GO
```

Uses

[Process]

Used By

[Report].[UspResultsRefresh_TableTimes]

Author: Johnson, Chris

[Process].[UspPopulate_HistoryTables1]

MS_Description

populates history tables from SysproTransactionsLogged

Parameters

Name	Data Type	Max Length (Bytes)
@RebuildBit	bit	1

SQL Script

```
CREATE Proc [Process].[UspPopulate_HistoryTables1] ( @RebuildBit Bit )
As /*
Template designed by Chris Johnson, Prometic Group September 2015
Stored procedure set out to query multiple databases with the same information and
return it in a collated format
--exec [Process].[UspPopulate_HistoryTables1] @RebuildBit =0
--exec [Process].[UspPopulate_HistoryTables1] @RebuildBit =1
*/
    Begin
--remove nocount on to speed up query

        Set NoCount On;

        Declare @ErrorMessage Varchar(50)= 'SQL transaction script is too long';
        Declare @CurrentTable Int= 1
          , @TotalTables Int
          , @SchemaName Varchar(500)
          , @TableName Varchar(500)
          , @NewTableName Varchar(500);
        Declare @Columns Varchar(500)
          , @Constraints Varchar(500);
        Declare @CurrentColumn Int = 1
          , @TotalColumns Int
          , @ColumnName Varchar(500)
          , @ColumnType Varchar(500);
        Declare @TablesToUpdate Varchar(Max);

--list the tables that are to be pulled back from each DB - if they are not found
the script will not be run against that db


--create temporary tables to be pulled from different databases, including a column
to id


--Table to capture all Transactions captured


        If Not Exists ( Select  [t].[name]
                        From    [sys].[tables] [t]
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Process.UspPopulate_-
HistoryTables1

```
                                Left Join [sys].[schemas] [s]
                                    On [s].[schema_id] = [t].[schema_id]
                        Where   [s].[name] = 'Process'
                                And [t].[name] = 'SysproTransactionsLogged' )
            Begin
                Create Table [Process].[SysproTransactionsLogged]
                    (
                      [TransactionDescription] Varchar(150)        Collate Latin1_-
General_BIN
                    , [DatabaseName] Varchar(150)                Collate Latin1_-
General_BIN
                    , [SignatureDateTime] As DateAdd(Millisecond ,
                                                     Cast(Substring(Cast([Signature-
Time] As Char(8)) ,
                                                                    7 , 2) As Int) ,
                                                     DateAdd(Second ,

Cast(Substring(Cast([SignatureTime] As Char(8)) ,
                                                                    5 , 2) As Int) ,
                                                                   DateAdd(Minute ,

Cast(Substring(Cast([SignatureTime] As Char(8)) ,
                                                                    3 , 2) As Int) ,
                                                                    DateAdd(Hour ,

Cast(Substring(Cast([SignatureTime] As Char(8)) ,
                                                                    1 , 2) As Int) ,
                                                                    Cast([SignatureDate]
As DateTime)))))
                    , [SignatureDate] Date
                    , [SignatureTime] Int
                    , [Operator] Varchar(20)                    Collate Latin1_-
General_BIN
                    , [VariableDesc] Varchar(100)                Collate Latin1_-
General_BIN
                    , [ItemKey] Varchar(150)                    Collate Latin1_-
General_BIN
                    , [VariableType] Char(1)
                    , [VarAlphaValue] Varchar(255)                Collate Latin1_-
General_BIN
                    , [VarNumericValue] Float
                    , [VarDateValue] DateTime2
                    , [ComputerName] Varchar(150)                Collate Latin1_-
General_BIN
                    , [ProgramName] Varchar(100)                Collate Latin1_-
General_BIN
                    , [TableName] Varchar(150)                    Collate Latin1_-
General_BIN
                    , [ConditionName] Varchar(15)                Collate Latin1_-
General_BIN
                    , [AlreadyEntered] Bit Default 0
                    , Constraint [TDR_AllKeys] Primary Key NonClustered
                        ( [DatabaseName] , [SignatureDate] , [SignatureTime] , [Item-
Key] , [Operator] , [ProgramName] , [VariableDesc] , [TableName] )
                        With ( Ignore_Dup_Key = On )
                    );
            End;
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Process.UspPopulate_-HistoryTables1

```
--create script to pull data from each db into the tables

        Declare @SQLTransactions Varchar(Max) = 'USE [?];
Declare @DB varchar(150),@DBCode varchar(150)
Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end
IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
BEGIN
BEGIN
Insert [BlackBox].[Process].[SysproTransactionsLogged] ([TransactionDescription],
[DatabaseName], [SignatureDate], [SignatureTime], [Operator], [VariableDesc], [Item-
Key], [VariableType], [VarAlphaValue], [VarNumericValue], [VarDateValue], [Computer-
Name], [ProgramName], [TableName], [ConditionName])
Select  [ATI].[TransactionDescription]
, Db_Name()
, [AD].[SignatureDate]
, [AD].[SignatureTime]
, [AD].[Operator]
, [VariableDesc] = Replace(Upper([AD].[VariableDesc]),'' '','''')
, [AL].[ItemKey]
, [AD].[VariableType]
, [VarAlphaValue] = Case When [AD].[VariableType] = ''A''
Then [AD].[VarAlphaValue]
Else Cast(Null As Varchar(255))
End
, [VarNumericValue] = Case When [AD].[VariableType] = ''N''
Then [AD].[VarNumericValue]
Else Cast(Null As Float)
End
, [VarDateValue] = Case When [AD].[VariableType] = ''D''
Then [AD].[VarDateValue]
Else Cast(Null As DateTime)
End
, [AL].[ComputerName]
, [AL].[ProgramName]
, [AL].[TableName]
, [AL].[ConditionName]
From    [dbo].[AdmSignatureLog] [AL]
Left Join [dbo].[AdmSignatureLogDet] [AD] On [AD].[TransactionId] =
[AL].[TransactionId]
And [AD].[SignatureDate] = [AL].[SignatureDate]
And [AD].[SignatureTime] = [AL].[SignatureTime]
And [AD].[SignatureLine] = [AL].[SignatureLine]
And [AD].[Operator] = [AL].[Operator]
Left Join [BlackBox].[Lookups].[AdmTransactionIDs] [ATI] On [ATI].[TransactionId] =
[AL].[TransactionId]
Where   [AL].[TableName] <> ''''
And [AD].[VariableDesc] <> ''''

Insert  [BlackBox].[Process].[SysproTransactionsLogged]
        ( [TransactionDescription]
        , [DatabaseName]
        , [SignatureDate]
        , [SignatureTime]
        , [Operator]
        , [VariableDesc]
        , [ItemKey]
```

Author: Johnson, Chris

```
        , [VariableType]
        , [VarAlphaValue]
        , [VarNumericValue]
        , [VarDateValue]
        , [ComputerName]
        , [ProgramName]
        , [TableName]
        , [ConditionName]
        )
        Select  [t].[TransactionDescription]
              , [t].[DatabaseName]
              , [t].[SignatureDate]
              , [t].[SignatureTime]
              , [t].[Operator]
              , [VariableDesc] = Replace(Upper([VariableDesc]),'' '','''')
              , [t].[ItemKey]
              , [t].[VariableType]
              , [t].[VarAlphaValue]
              , [t].[VarNumericValue]
              , [t].[VarDateValue]
              , [t].[ComputerName]
              , [t].[ProgramName]
              , [t].[TableName]
              , [t].[ConditionName]
        From    ( Select    [TransactionDescription] = Case When [ChangeFlag] =
''A''
                                                            Then ''Addition''
                                                            When [ChangeFlag] =
''U''
                                                            Then ''Update''
                                                            When [ChangeFlag] =
''X''
                                                            Then ''Deletion''
                                                            When [ChangeFlag] =
''C''
                                                            Then ''Change''
                                                            Else ''Unknown''
                                                       End
                          , [DatabaseName] = Db_Name()
                          , [SignatureDate] = [JnlDate]
                          , [SignatureTime] = [JnlTime]
                          , [Operator] = [OperatorCode]
                          , [VariableDesc] = [ColumnName] + ''Before''
                          , [ItemKey] = [Supplier]
                          , [VariableType] = ''A''
                          , [VarAlphaValue] = [Before]
                          , [VarNumericValue] = Null
                          , [VarDateValue] = Null
                          , [ComputerName] = ''''
                          , [ProgramName] = ''''
                          , [TableName] = ''ApSupplier''
                          , [ConditionName] = ''''
                  From      [dbo].[ApAmendmentJnl]
                  Union All
                  Select    [TransactionDescription] = Case When [ChangeFlag] =
''A''
                                                            Then ''Addition''
                                                            When [ChangeFlag] =
```

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Process.UspPopulate_-HistoryTables1

```
''U''
                                                        Then ''Update''
                                                        When [ChangeFlag] =
''X''
                                                        Then ''Deletion''
                                                        When [ChangeFlag] =
''C''
                                                        Then ''Change''
                                                        Else ''Unknown''
                                                    End
                        , Db_Name()
                        , [SignatureDate] = [JnlDate]
                        , [SignatureTime] = [JnlTime]
                        , [Operator] = [OperatorCode]
                        , [VariableDesc] = [ColumnName] + ''Before''
                        , [ItemKey] = [Currency]
                        , [VariableType] = ''A''
                        , [VarAlphaValue] = [Before]
                        , [VarNumericValue] = Null
                        , [VarDateValue] = Null
                        , [ComputerName] = ''''
                        , [ProgramName] = ''''
                        , [TableName] = ''TblCurrency''
                        , [ConditionName] = ''''
                From    [dbo].[TblCurAmendmentJnl]
                Union All
                Select  [TransactionDescription] = Case When [ChangeFlag] =
''A''
                                                        Then ''Addition''
                                                        When [ChangeFlag] =
''U''
                                                        Then ''Update''
                                                        When [ChangeFlag] =
''X''
                                                        Then ''Deletion''
                                                        When [ChangeFlag] =
''C''
                                                        Then ''Change''
                                                        Else ''Unknown''
                                                    End
                        , Db_Name()
                        , [SignatureDate] = [JnlDate]
                        , [SignatureTime] = [JnlTime]
                        , [Operator] = [OperatorCode]
                        , [VariableDesc] = [ColumnName] + ''Before''
                        , [ItemKey] = [Asset]
                        , [VariableType] = ''A''
                        , [VarAlphaValue] = [Before]
                        , [VarNumericValue] = Null
                        , [VarDateValue] = Null
                        , [ComputerName] = ''''
                        , [ProgramName] = ''''
                        , [TableName] = ''AssetMaster''
                        , [ConditionName] = ''''
                From    [dbo].[AssetAmendmentJnl]
                Union All
                Select  [TransactionDescription] = Case When [ChangeFlag] =
''A''
                                                        Then ''Addition''
```

Author: Johnson, Chris

```
                                                        When [ChangeFlag] =
''U''
                                                        Then ''Update''
                                                        When [ChangeFlag] =
''X''
                                                        Then ''Deletion''
                                                        When [ChangeFlag] =
''C''
                                                        Then ''Change''
                                                        Else ''Unknown''
                                                    End
                        , Db_Name()
                        , [SignatureDate] = [JnlDate]
                        , [SignatureTime] = [JnlTime]
                        , [Operator] = [Operator]
                        , [VariableDesc] = [ColumnName] + ''Before''
                        , [ItemKey] = [Supplier]
                        , [VariableType] = ''A''
                        , [VarAlphaValue] = [Before]
                        , [VarNumericValue] = Null
                        , [VarDateValue] = Null
                        , [ComputerName] = ''''
                        , [ProgramName] = ''''
                        , [TableName] = ''EftApSupplier''
                        , [ConditionName] = ''''
                from    [dbo].[EftCbAmendmentJnl]
                Union All
                Select  [TransactionDescription] = Case When [ChangeFlag] =
''A''
                                                        Then ''Addition''
                                                        When [ChangeFlag] =
''U''
                                                        Then ''Update''
                                                        When [ChangeFlag] =
''X''
                                                        Then ''Deletion''
                                                        When [ChangeFlag] =
''C''
                                                        Then ''Change''
                                                        Else ''Unknown''
                                                    End
                        , Db_Name()
                        , [SignatureDate] = [JnlDate]
                        , [SignatureTime] = [JnlTime]
                        , [Operator] = [OperatorName]
                        , [VariableDesc] = ''OldComVersion''
                        , [ItemKey] = [ParentPart] + ''_''
                          + Coalesce([Version] , '''') + ''_''
                          + Coalesce([SequenceNum] , '''') + ''_''
                          + Coalesce([Component] , '''')
                        , [VariableType] = ''A''
                        , [VarAlphaValue] = [OldComVersion]
                        , [VarNumericValue] = Null
                        , [VarDateValue] = Null
                        , [ComputerName] = ''''
                        , [ProgramName] = ''''
                        , [TableName] = ''BomStructure''
                        , [ConditionName] = ''''
                from    [dbo].[BomAmendmentJnl]
```

Author: Johnson, Chris

```
                  Union All
                  Select    [TransactionDescription] = Case When [ChangeFlag] =
''A''
                                                            Then ''Addition''
                                                            When [ChangeFlag] =
''U''
                                                            Then ''Update''
                                                            When [ChangeFlag] =
''X''
                                                            Then ''Deletion''
                                                            When [ChangeFlag] =
''C''
                                                            Then ''Change''
                                                            Else ''Unknown''
                                                       End
                          , Db_Name()
                          , [SignatureDate] = [JnlDate]
                          , [SignatureTime] = [JnlTime]
                          , [Operator] = [OperatorName]
                          , [VariableDesc] = ''OldComRelease''
                          , [ItemKey] = [ParentPart] + ''_''
                            + Coalesce([Version] , '''') + ''_''
                            + Coalesce([SequenceNum] , '''') + ''_''
                            + Coalesce([Component] , '''')
                          , [VariableType] = ''A''
                          , [VarAlphaValue] = [OldComRelease]
                          , [VarNumericValue] = Null
                          , [VarDateValue] = Null
                          , [ComputerName] = ''''
                          , [ProgramName] = ''''
                          , [TableName] = ''BomStructure''
                          , [ConditionName] = ''''
                  From      [dbo].[BomAmendmentJnl]
                  Union All
                  Select    [TransactionDescription] = Case When [ChangeFlag] =
''A''
                                                            Then ''Addition''
                                                            When [ChangeFlag] =
''U''
                                                            Then ''Update''
                                                            When [ChangeFlag] =
''X''
                                                            Then ''Deletion''
                                                            When [ChangeFlag] =
''C''
                                                            Then ''Change''
                                                            Else ''Unknown''
                                                       End
                          , [DatabaseName] = Db_Name()
                          , [SignatureDate] = [JnlDate]
                          , [SignatureTime] = [JnlTime]
                          , [Operator] = [OperatorCode]
                          , [VariableDesc] = [ColumnName] + ''Before''
                          , [ItemKey] = [StockCode]
                          , [VariableType] = ''A''
                          , [VarAlphaValue] = [Before]
                          , [VarNumericValue] = Null
                          , [VarDateValue] = Null
                          , [ComputerName] = ''''
```

Author: Johnson, Chris

```
                        , [ProgramName] = ''''
                        , [TableName] = ''InvMaster''
                        , [ConditionName] = ''''
                From    [dbo].[InvMastAmendJnl]
                Union   All
                Select  [TransactionDescription] = Case When [ChangeFlag] =
''A''
                                                    Then ''Addition''
                                                    When [ChangeFlag] =
''U''
                                                    Then ''Update''
                                                    When [ChangeFlag] =
''X''
                                                    Then ''Deletion''
                                                    When [ChangeFlag] =
''C''
                                                    Then ''Change''
                                                    Else ''Unknown''
                                                End
                        , [DatabaseName] = Db_Name()
                        , [SignatureDate] = [JnlDate]
                        , [SignatureTime] = [JnlTime]
                        , [Operator] = [OperatorCode]
                        , [VariableDesc] = [ColumnName] + ''Before''
                        , [ItemKey] = [Customer]
                        , [VariableType] = ''A''
                        , [VarAlphaValue] = [Before]
                        , [VarNumericValue] = Null
                        , [VarDateValue] = Null
                        , [ComputerName] = ''''
                        , [ProgramName] = ''''
                        , [TableName] = ''ArCustomer''
                        , [ConditionName] = ''''
                From    [dbo].[ArAmendmentJnl]
                Union   All
                Select  [TransactionDescription] = Case When [ChangeFlag] =
''A''
                                                    Then ''Addition''
                                                    When [ChangeFlag] =
''U''
                                                    Then ''Update''
                                                    When [ChangeFlag] =
''X''
                                                    Then ''Deletion''
                                                    When [ChangeFlag] =
''C''
                                                    Then ''Change''
                                                    Else ''Unknown''
                                                End
                        , [DatabaseName] = Db_Name()
                        , [SignatureDate] = [JnlDate]
                        , [SignatureTime] = [JnlTime]
                        , [Operator] = [OperatorCode]
                        , [VariableDesc] = [ColumnName] + ''Before''
                        , [ItemKey] = [Job]
                        , [VariableType] = ''A''
                        , [VarAlphaValue] = [Before]
                        , [VarNumericValue] = Null
                        , [VarDateValue] = Null
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Process.UspPopulate_-HistoryTables1

```
                        , [ComputerName] = ''''
                        , [ProgramName] = ''''
                        , [TableName] = ''WipMaster''
                        , [ConditionName] = ''''
                From    [dbo].[WipJobAmendJnl]
                Union   All
                Select  [TransactionDescription] = Case When [ChangeFlag] =
''A''
                                                        Then ''Addition''
                                                        When [ChangeFlag] =
''U''
                                                        Then ''Update''
                                                        When [ChangeFlag] =
''X''
                                                        Then ''Deletion''
                                                        When [ChangeFlag] =
''C''
                                                        Then ''Change''
                                                        Else ''Unknown''
                                                   End
                        , [DatabaseName] = Db_Name()
                        , [SignatureDate] = [JnlDate]
                        , [SignatureTime] = [JnlTime]
                        , [Operator] = [OperatorCode]
                        , [VariableDesc] = [ColumnName] + ''Before''
                        , [ItemKey] = [Warehouse]
                        , [VariableType] = ''A''
                        , [VarAlphaValue] = [Before]
                        , [VarNumericValue] = Null
                        , [VarDateValue] = Null
                        , [ComputerName] = ''''
                        , [ProgramName] = ''''
                        , [TableName] = ''WipMaster''
                        , [ConditionName] = ''''
                From    [dbo].[InvWhAmendJnl]
            ) [t];
        End
    End';

--Enable this function to check script changes (try to run script directly against
db manually)

        Print 'Length of Transaction script:'
            + Cast(Len(@SQLTransactions) As Varchar(50));
--Print @SQLTransactions


    --Remove previous transactions if being rebuilt, but only if table

        If @RebuildBit = 1
            Begin
                Print 'Rebuilt Option Selected - previous transactions removed';
                If Exists ( Select  [t].[name]
                            From    [sys].[tables] [t]
                                    Left Join [sys].[schemas] [s]
                                        On [s].[schema_id] = [t].[schema_id]
                            Where   [s].[name] = 'Process'
                                    And [t].[name] = 'SysproTransactionsLogged' )
```

Author: Johnson, Chris

```
                    Begin
                        Truncate Table [Process].[SysproTransactionsLogged];
                    End;
            End;

        --If Len(@SQLTransactions) <= 2000 --only run script if less than 2000

            --Begin

        Print 'Capturing Transactions';
        Exec [Process].[ExecForEachDB_WithTableCheck] @cmd = @SQLTransactions ,
            @SchemaTablesToCheck = N'AdmSignatureLog,AdmSignatureLogDet,ApAmendment-
Jnl,TblCurAmendmentJnl,AssetAmendmentJnl,EftCbAmendmentJnl,BomAmendmentJnl,InvMast-
AmendJnl,ArAmendmentJnl,WipJobAmendJnl,InvWhAmendJnl'; -- nvarchar(max)



            --End;

        --If Len(@SQLTransactions) > 2000 --if the script is greater than 2000 then
the script will probably fail - raise an error

            --Begin

                --Raiserror (@ErrorMessage,16,1); -- With Log;

            --End;


        If @RebuildBit = 1
            Begin

    --Get list of existing tables that use the History schema

                Create Table [#TablesToRename]
                    (
                      [TID] Int Identity(1 , 1)
                    , [SchemaName] Varchar(500) Collate Latin1_General_BIN
                    , [TableName] Varchar(500) Collate Latin1_General_BIN
                    , [NewTableName] As 'Archive' + [TableName]
                        + Upper(Replace(Replace(Convert(Varchar(24) , GetDate() ,
113) ,
                                                ' ' , '') , ':' , '')) --new table
name is old name plus the current timestamp

                    );

                Insert  [#TablesToRename]
                        ( [SchemaName]
                        , [TableName]
                        )
                        Select  [SchemaName] = [S].[name]
                              , [TableName] = [T].[name]
                        From    [sys].[schemas] As [S]
                                Left Join [sys].[tables] As [T]
                                    On [T].[schema_id] = [S].[schema_id]
                        Where   [S].[name] = 'History'
                                And [T].[name] Not Like 'Archive%'; --do not archive
tables that have already been archived;
```

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Process.UspPopulate_-HistoryTables1

```
    --Rename all existing tables


                --Find out how many tables need to be removed

                Select  @TotalTables = Max([TTR].[TID])
                From    [#TablesToRename] As [TTR];


                --advise user

                Print 'ExistingTables to Remove '
                    + Cast(@TotalTables As Varchar(50));

        --iterate through all tables and rename them


                If @TotalTables > 0
                    Begin
                        While @CurrentTable <= @TotalTables
                            Begin
                                Select  @TableName = [TTR].[TableName]
                                      , @SchemaName = [TTR].[SchemaName]
                                      , @NewTableName = [TTR].[NewTableName]
                                From    [#TablesToRename] As [TTR]
                                Where   [TTR].[TID] = @CurrentTable;

                                Exec [Process].[UspAlter_TableName] @SchemaName , --
varchar(500)

                                    @TableName , -- varchar(500)

                                    @NewTableName; -- varchar(500)

                                Set @CurrentTable = @CurrentTable + 1;
                            End;
                    End;

                Drop Table [#TablesToRename];

    --list of all tables to be created

                Create Table [#TablesToBeCreated]
                    (
                      [TID] Int Identity(1 , 1)
                    , [TableName] Varchar(500) Collate Latin1_General_BIN
                    );

                Insert  [#TablesToBeCreated]
                        ( [TableName]
                        )
                        Select Distinct
                                [TL].[TableName]
                        From    [BlackBox].[Process].[SysproTransactionsLogged]
```

Author: Johnson, Chris

```
                                As [TL];

    --Create all required tables with keys

                Set @CurrentTable = 1;
                Set @TotalTables = 0;

    --Get number of tables to create

                Select  @TotalTables = Max([TTBC].[TID])
                From    [#TablesToBeCreated] As [TTBC];

                Print 'Number of Tables to be created: '
                    + Cast(@TotalTables As Varchar(50));



                While @CurrentTable <= @TotalTables
                    Begin
                        Select  @SchemaName = 'History'
                              , @TableName = [TTBC].[TableName]
                        From    [#TablesToBeCreated] As [TTBC]
                        Where   [TTBC].[TID] = @CurrentTable;

                        Set @Columns = 'TransactionDescription VARCHAR(150)  Collate
Latin1_General_BIN, DatabaseName VARCHAR(150)  Collate Latin1_General_BIN, Signature-
DateTime DATETIME2, Operator VARCHAR(20)  Collate Latin1_General_BIN, ItemKey
VARCHAR(150)  Collate Latin1_General_BIN, ComputerName VARCHAR(150) Collate Latin1_-
General_BIN, ProgramName VARCHAR(100) Collate Latin1_General_BIN, ConditionName
VARCHAR(15)  Collate Latin1_General_BIN, AlreadyEntered BIT';
                        Set @Constraints = ' Constraint ' + @TableName
                            + '_AllKeys Primary Key NonClustered ( DatabaseName,
SignatureDateTime, ItemKey, Operator, ProgramName ) With ( Ignore_Dup_Key = On )';

                        Exec [Process].[UspCreate_Table] @SchemaName , --
varchar(500)

                            @TableName , -- varchar(500)

                            @Columns , -- varchar(500)

                            @Constraints;-- varchar(500)


                        Print 'Table ' + @TableName + ' '
                            + Cast(@CurrentTable As Varchar(50)) + ' created';


                        Set @CurrentTable = @CurrentTable + 1;
                    End;

                Drop Table [#TablesToBeCreated];

                Exec [Process].[UspAlter_CheckAndAddColumns];


            End;
        If @RebuildBit = 0
            Begin
```

```
                Print 'Rebuild option not selected';
            --Get list of tables that don't exist

                Create Table [#MissingTables]
                    (
                      [TID] Int Identity(1 , 1)
                    , [TableName] Varchar(500) Collate Latin1_General_BIN
                    );

                Insert  [#MissingTables]
                        ( [TableName]
                        )
                        Select Distinct
                                [TL].[TableName]
                        From    [BlackBox].[Process].[SysproTransactionsLogged]
                                As [TL]
                        Where   [TL].[TableName] Not In (
                                Select  [T].[name]
                                From    [sys].[schemas] As [S]
                                        Left Join [sys].[tables] [T]
                                            On [T].[schema_id] = [S].[schema_id]
                                Where   [S].[name] = 'History' );

    --Create all required tables with keys

                Set @CurrentTable = 1;
                Set @TotalTables = 0;

    --Get number of tables to create

                Select  @TotalTables = Max([TTBC].[TID])
                From    [#MissingTables] As [TTBC];

                Print 'Number of Tables to be created: '
                    + Cast(@TotalTables As Varchar(50));



                While @CurrentTable <= @TotalTables
                    Begin
                        Select  @SchemaName = 'History'
                              , @TableName = [TTBC].[TableName]
                        From    [#MissingTables] As [TTBC]
                        Where   [TTBC].[TID] = @CurrentTable;

                        Set @Columns = 'TransactionDescription VARCHAR(150),
DatabaseName VARCHAR(150), SignatureDateTime DATETIME2, Operator VARCHAR(20), Item-
Key VARCHAR(150), ComputerName VARCHAR(150), ProgramName VARCHAR(100), ConditionName
VARCHAR(15), AlreadyEntered BIT';
                        Set @Constraints = ' Constraint ' + @TableName
                            + '_AllKeys Primary Key NonClustered ( DatabaseName,
SignatureDateTime, ItemKey, Operator, ProgramName ) With ( Ignore_Dup_Key = On )';

                        Exec [Process].[UspCreate_Table] @SchemaName , --
varchar(500)

                            @TableName , -- varchar(500)
```

Author: Johnson, Chris

```
                            @Columns , -- varchar(500)

                            @Constraints;-- varchar(500)


                        Print 'Table ' + @TableName + ' '
                            + Cast(@CurrentTable As Varchar(50)) + ' created';


                        Set @CurrentTable = @CurrentTable + 1;
                    End;

                Drop Table [#MissingTables];

                Exec [Process].[UspAlter_CheckAndAddColumns];
            End;

        --run insert

        Select  @TablesToUpdate = Stuff(( Select Distinct
                                                ', '
                                                + Cast(''
                                                + [STL].[TableName] + '' As
Varchar(150))
                                          From      [Process].[SysproTransactions-
Logged]
                                                    As [STL]
                                          Where     [STL].[AlreadyEntered] = 0
                                        For
                                          Xml Path('')
                                        ) , 1 , 1 , '');

        Print 'Tables to update ' + @TablesToUpdate;

        Exec [Process].[UspPopulate_UnpivotHistory] @Tables = @TablesToUpdate;
    End;


GO
EXEC sp_addextendedproperty N'MS_Description', N'populates history tables from
SysproTransactionsLogged', 'SCHEMA', N'Process', 'PROCEDURE', N'UspPopulate_History-
Tables1', NULL, NULL
GO
```

Uses

[Process].[SysproTransactionsLogged]
[Process].[ExecForEachDB_WithTableCheck]
[Process].[UspAlter_CheckAndAddColumns]
[Process].[UspAlter_TableName]
[Process].[UspCreate_Table]
[Process].[UspPopulate_UnpivotHistory]
[Process]

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Process.UspPopulate_-HistoryTables1

Used By

[Report].[UspResultsRefresh_TableTimes]

Author: Johnson, Chris

[Process].[UspPopulate_UnpivotHistory]

MS_Description

Stored proc to update specified tables, creating new fields on the fly

Parameters

Name	Data Type	Max Length (Bytes)
@Tables	varchar(max)	max

SQL Script

```
CREATE Proc [Process].[UspPopulate_UnpivotHistory] ( @Tables Varchar(Max) )
As
    Begin
--List of Tables insert into

        If Replace(@Tables , ' ' , '') <> ''
            Begin
                Create Table [#TablesToPopulate]
                    (
                      [TID] Int Identity(1 , 1)
                    , [TableName] Varchar(500) Collate Latin1_General_BIN
                    );

                Insert  [#TablesToPopulate]
                        ( [TableName]
                        )
                        Select Distinct
                                Replace([USS].[Value] , ' ' , '')
                        From    [dbo].[udf_SplitString](@Tables , ',') As [USS];


                Create Table [#Transactions]
                    (
                      [TransactionDescription] Varchar(500) Collate Latin1_General_-
BIN
                    , [DatabaseName] Varchar(255)            Collate Latin1_General_-
BIN
                    , [SignatureDateTime] DateTime
                    , [Operator] Varchar(255)                Collate Latin1_General_-
BIN
                    , [ItemKey] Varchar(500)                Collate Latin1_General_-
BIN
                    , [ComputerName] Varchar(500)            Collate Latin1_General_-
BIN
                    , [ProgramName] Varchar(500)            Collate Latin1_General_-
BIN
                    , [ConditionName] Varchar(500)            Collate Latin1_-
General_BIN
                    , [VariableDesc] Varchar(500)            Collate Latin1_General_-
BIN
                    , [VariableRank] Int
```

Author: Johnson, Chris

```
                    , [TransactionRank] Int
                    );


                Declare @MaxTables Int
                  , @CurrentTable Int = 1
                  , @SQLPivot Varchar(Max)
                  , @Columns Varchar(Max)
                  , @TableName Varchar(500)
                  , @MaxTransactions Int
                  , @CurrentTransaction Int = 1
                  , @MaxVariables Int
                  , @CurrentVariable Int = 1
                  , @VarDate Date
                  , @VarString Varchar(255)
                  , @VarNum Float
                  , @VarName Varchar(500)
                  , @VarDbName Varchar(500)
                  , @VarItemKey Varchar(500)
                  , @VarSignatureDateTime DateTime
                  , @SQLUpdate Varchar(Max)
                  , @SQLInsert Varchar(Max);

                Select  @MaxTables = Max([TTP].[TID])
                From    [#TablesToPopulate] As [TTP];
                Print Cast(@MaxTables As Varchar(50)) + ' tables to unpivot';

                While @CurrentTable <= @MaxTables
                    Begin
                        Select  @TableName = [TTP].[TableName]
                        From    [#TablesToPopulate] As [TTP]
                        Where   [TTP].[TID] = @CurrentTable;

                        Print 'Unpivotting table ' + @TableName + ' '
                            + Cast(@CurrentTable As Varchar(50));

                        Select  @Columns = Stuff(( Select Distinct
                                                          ', '
                                                          + Cast('['
                                                          + [STL].[VariableDesc]
                                                          + ']' As Varchar(150))
                                                   From     [Process].[Syspro-
TransactionsLogged]
                                                            As [STL]
                                                   Where    [STL].[TableName] =
@TableName
                                                   Order By ( ', '
                                                              + Cast('['
                                                              + [STL].[VariableDesc]
                                                              + ']' As Varchar(150))
) Asc
                                                 For
                                                   Xml Path('')
                                                 ) , 1 , 1 , '');


--Insert All transactions
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Process.UspPopulate_-UnpivotHistory

```
                        Set @SQLInsert = 'Insert [History].' + @TableName
                            + ' ( [TransactionDescription], [DatabaseName],
[SignatureDateTime], [Operator], [ItemKey], [ComputerName], [ProgramName],
[ConditionName])
SELECT [TransactionDescription], [DatabaseName], [SignatureDateTime], [Operator],
[ItemKey], [ComputerName], [ProgramName], [ConditionName]
From [Process].[SysproTransactionsLogged] As [STL]
Where [STL].[TableName]=''' + @TableName + '''
and [AlreadyEntered]=0';

                        Exec ( @SQLInsert );


                        Insert  [#Transactions]
                                ( [TransactionDescription]
                                , [DatabaseName]
                                , [SignatureDateTime]
                                , [Operator]
                                , [ItemKey]
                                , [ComputerName]
                                , [ProgramName]
                                , [ConditionName]
                                , [VariableDesc]
                                , [VariableRank]
                                , [TransactionRank]
                                )
                                Select Top 10000 --limit to 10,000 at a time

                                        [STL].[TransactionDescription]
                                      , [STL].[DatabaseName]
                                      , [STL].[SignatureDateTime]
                                      , [STL].[Operator]
                                      , [STL].[ItemKey]
                                      , [STL].[ComputerName]
                                      , [STL].[ProgramName]
                                      , [STL].[ConditionName]
                                      , [STL].[VariableDesc]
                                      , [VariableRank] = Dense_Rank() Over (
Partition By [STL].[TransactionDescription] ,
                                                                            [STL].[DatabaseName] ,
                                                                            [STL].[SignatureDate-
Time] ,
                                                                            [STL].[Operator] ,
                                                                            [STL].[ItemKey] ,
                                                                            [STL].[ComputerName] ,
                                                                            [STL].[ProgramName] ,
                                                                            [STL].[ConditionName]
Order By [STL].[VariableDesc] )
                                      , [TransactionRank] = Dense_Rank() Over (
Order By [STL].[TransactionDescription], [STL].[DatabaseName], [STL].[SignatureDate-
Time], [STL].[Operator], [STL].[ItemKey], [STL].[ComputerName], [STL].[ProgramName],
[STL].[ConditionName] )
                                FROM    [Process].[SysproTransactionsLogged]
                                        As [STL]
                                Where   [STL].[TableName] = @TableName
                                        And [STL].[AlreadyEntered] = 0;

                        Select  @MaxTransactions = Max([T].[TransactionRank])
                        From    [#Transactions] As [T];
```

Author: Johnson, Chris

```
                        Set @CurrentTransaction = 1;

                        While @CurrentTransaction <= @MaxTransactions
                            Begin
                                Select  @MaxVariables = Max([T].[VariableRank])
                                From    [#Transactions] As [T]
                                Where   [T].[TransactionRank] = @CurrentTransaction;
                                Set @CurrentVariable = 1;
                                While @CurrentVariable <= @MaxVariables
                                    Begin
                                        Set @VarDate = Null;
                                        Set @VarNum = Null;
                                        Set @VarString = Null;
                                        Set @VarName = Null;

                                        Select  @VarName = [T].[VariableDesc]
                                              , @VarNum = [STL].[VarNumericValue]
                                              , @VarString = Replace([STL].[VarAlpha-
Value] ,
                                                                     '''' , '''''')
                                              , @VarDate = [STL].[VarDateValue]
                                              , @VarDbName = [STL].[DatabaseName]
                                              , @VarItemKey = [STL].[ItemKey]
                                              , @VarSignatureDateTime =
[STL].[SignatureDateTime]
                                        From    [#Transactions] As [T]
                                                Left Join [Process].[Syspro-
TransactionsLogged]
                                                    As [STL]
                                                    On [STL].[DatabaseName] =
[T].[DatabaseName]
                                                       And [STL].[ItemKey] =
[T].[ItemKey]
                                                       And [STL].[SignatureDateTime]
= [T].[SignatureDateTime]
                                                       And [STL].[VariableDesc] =
[T].[VariableDesc]
                                        Where   [T].[TransactionRank] = @Current-
Transaction
                                                And [T].[VariableRank] = @Current-
Variable;
                                        If Coalesce(@VarString ,
                                                    Convert(Varchar(500) , @VarNum)
,
                                                    Convert(Varchar(500) , @Var-
Date)) Is Not Null
                                            Begin
                                                Set @SQLUpdate = 'Begin Try
                                    Update [History].' + QuoteName(@TableName) + '
        Set ' + QuoteName(@VarName) + ' = '
                                                    + Case When @VarNum Is Not Null
                                                           Then 'Cast('''
                                                              + Cast(@VarNum As
Varchar(500))
                                                              + '''as float)'
                                                           When @VarString Is Not
Null
                                                           Then ''''
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Process.UspPopulate_-UnpivotHistory

```
                                                                + @VarString
                                                                + ''''
                                                            When @VarDate Is Not Null
                                                            Then 'Cast('''
                                                                + Cast(@VarDate As
Varchar(500))
                                                                + '''as date)
                                                    '
                                                      End
                                                + 'Where [DatabaseName]='''
                                                + @VarDbName
                                                + ''' And [ItemKey] ='''
                                                + @VarItemKey
                                                + ''' And CONVERT(VARCHAR(23),
[SignatureDateTime], 121)='''
                                                + Convert(Varchar(23) , @Var-
SignatureDateTime , 121)
                                                + '''
                                End Try
Begin Catch
    Update  [Process].[SysproTransactionsLogged]
    Set     [IsError] = 1
    Where   [DatabaseName] = ''' + @VarDbName + '''
            And [ItemKey] = ''' + @VarItemKey + '''
            And CONVERT(VARCHAR(23), [SignatureDateTime], 121)='''
                                                + Convert(Varchar(23) , @Var-
SignatureDateTime , 121)
                                                + '''
            And [VariableDesc] = ''' + @VarName + ''';
End Catch';

                                            Exec ( @SQLUpdate );
                                        End;
                                    Update  [Process].[SysproTransactionsLogged]
                                    Set     [AlreadyEntered] = 1
                                    From    [Process].[SysproTransactionsLogged]
[STL]
                                            Inner Join [#Transactions] [T]
                                                On [STL].[DatabaseName] =
[T].[DatabaseName]
                                                   And [STL].[ItemKey] =
[T].[ItemKey]
                                                   And [STL].[SignatureDateTime]
= [T].[SignatureDateTime]
                                                   And [STL].[VariableDesc] =
[T].[VariableDesc]
                                    Where   [T].[TransactionRank] = @Current-
Transaction
                                            And [T].[VariableRank] = @Current-
Variable;


                                    Set @CurrentVariable = @CurrentVariable
                                        + 1;
                                End;
                            Set @CurrentTransaction = @CurrentTransaction
                                + 1;
                        End;

                    Print @TableName + ' unpivotted';
```

Author: Johnson, Chris

```
                        Truncate Table [#Transactions];
                        Set @CurrentTable = @CurrentTable + 1;
                    End;
            End;
    End;
GO
EXEC sp_addextendedproperty N'MS_Description', N'Stored proc to update specified
tables, creating new fields on the fly', 'SCHEMA', N'Process', 'PROCEDURE', N'Usp-
Populate_UnpivotHistory', NULL, NULL
GO
```

Uses

[Process].[SysproTransactionsLogged]
[dbo].[udf_SplitString]
[Process]

Used By

[Process].[UspPopulate_HistoryTables1]

[Process].[UspUpdate_ApSupplier]

MS_Description

Stored proc to update specified table

Parameters

Name	Data Type	Max Length (Bytes)
@PrevCheck	int	4
@HoursBetweenUpdates	numeric(5,2)	5

SQL Script

```
CREATE Proc [Process].[UspUpdate_ApSupplier]
    (
      @PrevCheck Int
    , @HoursBetweenUpdates Numeric(5 , 2)
    )
As
    Begin
--remove nocount on to speed up query

        Set NoCount On;

--check if table exists and create if it doesn't

        If ( Not Exists ( Select    1
                          From      [INFORMATION_SCHEMA].[TABLES]
                          Where     [TABLE_SCHEMA] = 'Lookups'
                                    And [TABLE_NAME] = 'ApSupplier' )
           )
            Begin
                Create Table [Lookups].[ApSupplier]
                    (
                      [Company] Varchar(150) Collate Latin1_General_BIN
                    , [Supplier] Varchar(150) Collate Latin1_General_BIN
                    , [SupplierName] Varchar(150) Collate Latin1_General_BIN
                    , [LastUpdated] DateTime2
                    , [ActivePOFlag] Bit
                    );
            End;

--check last time run and update if it's been longer than @HoursBetweenUpdates hours

        Declare @LastDate DateTime2;

        Select  @LastDate = Max([LastUpdated])
        From    [Lookups].[ApSupplier];

        If @LastDate Is Null
            Or DateDiff(Minute , @LastDate , GetDate()) > ( @HoursBetweenUpdates
```

Author: Johnson, Chris

```
                                                        * 60 )
            Begin
    --Set time of run

                Declare @LastUpdated DateTime2;
                Select  @LastUpdated = GetDate();

--list the tables that are to be pulled back from each DB - if they are not found
the script will not be run against that db

                Declare @ListOfTables Varchar(Max) = 'ApSupplier';

--create temporary tables to be pulled from different databases, including a column
to id

                Create Table [#Table1Supplier]
                    (
                      [Company] Varchar(150) Collate Latin1_General_BIN
                    , [Supplier] Varchar(150) Collate Latin1_General_BIN
                    , [SupplierName] Varchar(150) Collate Latin1_General_BIN
                    , [ActivePOFlag] Bit
                    );

--create script to pull data from each db into the tables

                Declare @SQL Varchar(Max) = 'USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
            BEGIN
                Insert #Table1Supplier
                    ( Company, Supplier, [SupplierName], [ActivePOFlag])
                Select @DBCode
                ,[APS].Supplier
                ,[APS].[SupplierName]
                , [ActivePOFlag] = Max(Case When [PMD].[PurchaseOrder] Is Null Then
0
                                Else 1
                            End)
From    [dbo].[ApSupplier] [APS]
        Left Join [dbo].[PorMasterHdr] [PMH]
            On [PMH].[Supplier] = [APS].[Supplier]
               And [PMH].[CancelledFlag] <> ''Y''
               And [PMH].[DatePoCompleted] Is Null
        Left Join [dbo].[PorMasterDetail] [PMD]
            On [PMD].[PurchaseOrder] = [PMH].[PurchaseOrder]
               And [PMD].[MOrderQty] <> [PMD].[MReceivedQty]
Group By [APS].[Supplier]
      , [APS].[SupplierName];
End
';

--execute script against each db, populating the base tables

                Exec [Process].[ExecForEachDB_WithTableCheck] @cmd = @SQL ,
                    @SchemaTablesToCheck = @ListOfTables;
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Process.UspUpdate_Ap-Supplier

```
                Insert  [Lookups].[ApSupplier]
                        ( [Company]
                        , [Supplier]
                        , [LastUpdated]
                        , [SupplierName]
                        , [ActivePOFlag]
                        )
                        Select  [Company]
                              , [Supplier]
                              , @LastUpdated
                              , [SupplierName]
                              , [ActivePOFlag]
                        From    [#Table1Supplier];

                If @PrevCheck = 1
                    Begin
                        Declare @CurrentCount Int
                          , @PreviousCount Int;

                        Select  @CurrentCount = Count(*)
                        From    [Lookups].[ApSupplier]
                        Where   [LastUpdated] = @LastUpdated;

                        Select  @PreviousCount = Count(*)
                        From    [Lookups].[ApSupplier]
                        Where   [LastUpdated] <> @LastUpdated;

                        If @PreviousCount > @CurrentCount
                            Begin
                                Delete  [Lookups].[ApSupplier]
                                Where   [LastUpdated] = @LastUpdated;
                                Print 'UspUpdate_ApSupplier - Count has gone down
since last run, no update applied';
                                Print 'Current Count = '
                                    + Cast(@CurrentCount As Varchar(5))
                                    + ' Previous Count = '
                                    + Cast(@PreviousCount As Varchar(5));
                            End;
                        If @PreviousCount <= @CurrentCount
                            Begin
                                Delete  [Lookups].[ApSupplier]
                                Where   [LastUpdated] <> @LastUpdated;
                                Print 'UspUpdate_ApSupplier - Update applied
successfully';
                            End;
                    End;
                If @PrevCheck = 0
                    Begin
                        Delete  [Lookups].[ApSupplier]
                        Where   [LastUpdated] <> @LastUpdated;
                        Print 'UspUpdate_ApSupplier - Update applied successfully';
                    End;
            End;
    End;
    If DateDiff(Minute , @LastDate , GetDate()) <= ( @HoursBetweenUpdates * 60 )
        Begin
            Print 'UspUpdate_ApSupplier - Table was last updated at '
```

Author: Johnson, Chris

```
                + Cast(@LastDate As Varchar(255)) + ' no update applied';
        End;



GO
EXEC sp_addextendedproperty N'MS_Description', N'Stored proc to update specified
table', 'SCHEMA', N'Process', 'PROCEDURE', N'UspUpdate_ApSupplier', NULL, NULL
GO
```

Uses

[Lookups].[ApSupplier]

[Process].[ExecForEachDB_WithTableCheck]

[Process]

[Process].[UspUpdate_ArInvoicePayTrnType]

MS_Description

Stored proc to update specified table

Parameters

Name	Data Type	Max Length (Bytes)
@PrevCheck	int	4
@HoursBetweenUpdates	numeric(5,2)	5

SQL Script

```
CREATE Proc [Process].[UspUpdate_ArInvoicePayTrnType]
    (
      @PrevCheck Int
    , @HoursBetweenUpdates Numeric(5 , 2)
    )
As
    Begin

--check if table exists and create if it doesn't

        If ( Not Exists ( Select    *
                          From      [INFORMATION_SCHEMA].[TABLES]
                          Where     [TABLE_SCHEMA] = 'Lookups'
                                    And [TABLE_NAME] = 'ArInvoicePayTrnType' )
           )
            Begin
                Create Table [Lookups].[ArInvoicePayTrnType]
                    (
                      [TrnType] Char(1) Collate Latin1_General_BIN
                    , [TrnTypeDesc] Varchar(250) Collate Latin1_General_BIN
                    , [LastUpdated] DateTime2
                    );
            End;


        Declare @LastUpdate DateTime2 = GetDate();
        Declare @LastDate DateTime2;

        Select  @LastDate = Max([bt].[LastUpdated])
        From    [Lookups].[ArInvoicePayTrnType] As [bt];
        If DateDiff(Minute , @LastDate , GetDate()) > ( @HoursBetweenUpdates
                                                        * 60 )
            Begin
                Insert  [Lookups].[ArInvoicePayTrnType]
                        ( [TrnType]
                        , [TrnTypeDesc]
                        , [LastUpdated]
                        )
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Process.UspUpdate_Ar-InvoicePayTrnType

```
                        Select  [t].[TrnType]
                              , [t].[TrnTypeDesc]
                              , [LastUpdated] = @LastUpdate
                        From    ( Select    [TrnType] = 'A'
                                          , [TrnTypeDesc] = 'Adjustment'
                                  Union
                                  Select    [TrnType] = 'C'
                                          , [TrnTypeDesc] = 'Credit memo'
                                  Union
                                  Select    [TrnType] = 'D'
                                          , [TrnTypeDesc] = 'Debit memo'
                                  Union
                                  Select    [TrnType] = 'P'
                                          , [TrnTypeDesc] = 'Payment'
                                  Union
                                  Select    [TrnType] = 'V'
                                          , [TrnTypeDesc] = 'Exchange rate
revaluation'
                                  Union
                                  Select    [TrnType] = 'T'
                                          , [TrnTypeDesc] = 'Tax relief adjustment'
                                ) [t];

                If @PrevCheck = 1
                    Begin
                        Declare @CurrentCount Int
                          , @PreviousCount Int;

                        Select  @CurrentCount = Count(*)
                        From    [Lookups].[ArInvoicePayTrnType]
                        Where   [LastUpdated] = @LastUpdate;

                        Select  @PreviousCount = Count(*)
                        From    [Lookups].[ArInvoicePayTrnType]
                        Where   [LastUpdated] <> @LastUpdate;

                        If @PreviousCount > @CurrentCount
                            Begin
                                Delete  [Lookups].[ArInvoicePayTrnType]
                                Where   [LastUpdated] = @LastUpdate;
                                Print 'UspUpdate_ArInvoicePayTrnType - Count has
gone down since last run, no update applied';
                                Print 'Current Count = '
                                    + Cast(@CurrentCount As Varchar(5))
                                    + ' Previous Count = '
                                    + Cast(@PreviousCount As Varchar(5));
                            End;
                        If @PreviousCount <= @CurrentCount
                            Begin
                                Delete  [Lookups].[ArInvoicePayTrnType]
                                Where   [LastUpdated] <> @LastUpdate;
                                Print 'UspUpdate_ArInvoicePayTrnType - Update
applied successfully';
                            End;
                    End;
                If @PrevCheck = 0
                    Begin
                        Delete  [Lookups].[ArInvoicePayTrnType]
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Process.UspUpdate_Ar-InvoicePayTrnType

```
                        Where   [LastUpdated] <> @LastUpdate;
                        Print 'UspUpdate_ArInvoicePayTrnType - Update applied
successfully';
                    End;
            End;
        If DateDiff(Minute , @LastDate , GetDate()) <= ( @HoursBetweenUpdates
                                                        * 60 )
            Begin
                Print 'UspUpdate_ArInvoicePayTrnType - Table was last updated at '
                    + Cast(@LastDate As Varchar(255)) + ' no update applied';
            End;
    End;
GO
EXEC sp_addextendedproperty N'MS_Description', N'Stored proc to update specified
table', 'SCHEMA', N'Process', 'PROCEDURE', N'UspUpdate_ArInvoicePayTrnType', NULL,
NULL
GO
```

Uses

[Lookups].[ArInvoicePayTrnType]
[Process]

Author: Johnson, Chris

[Process].[UspUpdate_BankBalances]

MS_Description

Stored proc to update specified table

Parameters

Name	Data Type	Max Length (Bytes)
@PrevCheck	int	4
@HoursBetweenUpdates	numeric(6,2)	5

SQL Script

```
CREATE Proc [Process].[UspUpdate_BankBalances]
    (
      @PrevCheck Int
    , @HoursBetweenUpdates Numeric(6,2)
    )
As
    Begin
/*
Stored procedure created by Chris Johnson, Prometic Group September 2015 to populate
table with BankBalances details
transaction types when relating to inventory changes
--Exec [Process].[UspUpdate_BankBalances]@PrevCheck =0    , @HoursBetweenUpdates =0
*/

--remove nocount on to speed up query

        Set NoCount On;

        Declare @LoadDate Date = GetDate();
--Create Lookup table if it doesn't exist

        If ( Not Exists ( Select    1
                          From      [INFORMATION_SCHEMA].[TABLES]
                          Where     [TABLE_SCHEMA] = 'Lookups'
                                    And [TABLE_NAME] = 'BankBalances' )
           )
            Begin

                Create Table [Lookups].[BankBalances]
                    (
                      [DatabaseName] Varchar(150) Collate Latin1_General_BIN
                    , [CompanyName] Varchar(150) Collate Latin1_General_BIN
                    , [Bank] Varchar(10) Collate Latin1_General_BIN
                    , [BankDescription] Varchar(150) Collate Latin1_General_BIN
                    , [CashGlCode] Varchar(150) Collate Latin1_General_BIN
                    , [BankCurrency] Char(3) Collate Latin1_General_BIN
                    , [CurrentBalance] Numeric(20 , 7)
                    , [StatementBalance] Numeric(20 , 7)
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Process.UspUpdate_-BankBalances

```
                    , [OutStandingDeposits] Numeric(20 , 7)
                    , [OutStandingWithdrawals] Numeric(20 , 7)
                    , [PrevMonth1CurrentBalance] Numeric(20 , 7)
                    , [PrevMonth1StatementBalance] Numeric(20 , 7)
                    , [PrevMonth1OutStandingDeposits] Numeric(20 , 7)
                    , [PrevMonth1OutStandingWithdrawals] Numeric(20 , 7)
                    , [PrevMonth2CurrentBalance] Numeric(20 , 7)
                    , [PrevMonth2StatementBalance] Numeric(20 , 7)
                    , [PrevMonth2OutStandingDeposits] Numeric(20 , 7)
                    , [PrevMonth2OutStandingWithdrawals] Numeric(20 , 7)
                    , [DateOfBalance] Date
                    , [DateTimeOfBalance] DateTime2 Default GetDate()
                    );
            End;

--remove any balances loaded already today

        Delete  From [Lookups].[BankBalances]
        Where   [DateOfBalance] = @LoadDate;

        Declare @ListOfTables Varchar(Max) = 'ApBank';

--create script to pull data from each db into the tables

        Declare @Company Varchar(max) = 'All';
        Declare @SQLBanks Varchar(Max) = 'USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end'
            + --Only query DBs beginning SysProCompany

            '
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN'
            + --only companies selected in main run, or if companies selected then
all

            '
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT'
            + --count number of tables requested (number of commas plus one)

            '
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')'
            + --Count of the tables requested how many exist in the db

            '
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_Bin From Black-
Box.dbo.udf_SplitString(@ListOfTables,'',''))  '
            + --only if the count matches (all the tables exist in the requested db)
then run the script

            '
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Process.UspUpdate_-BankBalances

```
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
            Declare @LD date =''' + Cast(@LoadDate As Varchar(50))
            + '''
                Insert  [BlackBox].[Lookups].[BankBalances]
        ( [DatabaseName], [CompanyName], [Bank], [BankDescription], [CashGlCode],
[BankCurrency], [CurrentBalance], [StatementBalance], [OutStandingDeposits], [Out-
StandingWithdrawals], [PrevMonth1CurrentBalance], [PrevMonth1StatementBalance],
[PrevMonth1OutStandingDeposits], [PrevMonth1OutStandingWithdrawals], [Prev-
Month2CurrentBalance], [PrevMonth2StatementBalance], [PrevMonth2OutStanding-
Deposits], [PrevMonth2OutStandingWithdrawals], [DateOfBalance])
         Select  Db_Name()
                        , [CN].[CompanyName]
                        , [AP].[Bank]
                        , [AP].[Description]
                        , [AP].[CashGlCode]
                        , [AP].[Currency]
                        , [AP].[CbCurBalLoc1]
                        , [AP].[CbStmtBal1]
                        , [AP].[OutstDep1]
                        , [AP].[OutstWith1]
                        , [AP].[CbCurBal2]
                        , [AP].[CbStmtBal2]
                        , [AP].[OutstDep2]
                        , [AP].[OutstWith2]
                        , [AP].[CbCurBal3]
                        , [AP].[CbStmtBal3]
                        , [AP].[OutstDep3]
                        , [AP].[OutstWith3]
                        , @LD
                    From    [dbo].[ApBank] [AP]
                            Cross Join [BlackBox].[Lookups].[CompanyNames] As [CN]
                    Where   [CN].[Company] = @DBCode;
            End
    End';

--Enable this function to check script changes (try to run script directly against
db manually)

--Print @SQLBanks


--execute script against each db, populating the base tables

        Exec [Process].[ExecForEachDB] @cmd = @SQLBanks;

        If ( Select Count(1)
             From   [Lookups].[BankBalances] As [BB]
             Where  [BB].[DateOfBalance] = Cast(GetDate() As Date)
           ) > 1
            Begin
                Print 'UspUpdate_BankBalances - Update applied successfully';
            End;
        If ( Select Count(1)
             From   [Lookups].[BankBalances] As [BB]
             Where  [BB].[DateOfBalance] = Cast(GetDate() As Date)
           ) = 0
            Begin
                Declare @ErrorMessage Varchar(150)= 'UspUpdate_BankBalances - Update
```

Author: Johnson, Chris

```
Error';
                Raiserror (@ErrorMessage,16,1);
            End;
    End;


GO
EXEC sp_addextendedproperty N'MS_Description', N'Stored proc to update specified
table', 'SCHEMA', N'Process', 'PROCEDURE', N'UspUpdate_BankBalances', NULL, NULL
GO
```

Uses

[Lookups].[BankBalances]

[Process].[ExecForEachDB]

[Process]

[Process].[UspUpdate_Bin]

MS_Description

Stored proc to update specified table

Parameters

Name	Data Type	Max Length (Bytes)
@PrevCheck	int	4
@HoursBetweenUpdates	numeric(5,2)	5

SQL Script

```
CREATE Proc [Process].[UspUpdate_Bin]
    (
      @PrevCheck Int
    , @HoursBetweenUpdates Numeric(5,2)
    )
As
    Begin
/*
Stored procedure created by Chris Johnson, Prometic Group September 2015 to populate
table with Bin details    transaction types when relating to inventory changes
*/

--remove nocount on to speed up query

        Set NoCount On;

--check if table exists and create if it doesn't

        If ( Not Exists ( Select    *
                          From      [INFORMATION_SCHEMA].[TABLES]
                          Where     [TABLE_SCHEMA] = 'Lookups'
                                    And [TABLE_NAME] = 'Bin' )
           )
            Begin
                Create Table [Lookups].[Bin]
                    (
                      [Company] Varchar(150)    collate Latin1_General_BIN
                    , [Bin] Varchar(150)        collate Latin1_General_BIN
                    , [LastUpdated] DateTime2
                    );
            End;



--check last time run and update if it's been longer than @HoursBetweenUpdates hours

        Declare @LastDate DateTime2;
```

Author: Johnson, Chris

```
        Select  @LastDate = Max([LastUpdated])
        From    [Lookups].[Bin];

        If @LastDate Is Null
            Or DateDiff(Minute , @LastDate , GetDate()) > (@HoursBetweenUpdates*60)
            Begin
    --Set time of run

                Declare @LastUpdated DateTime2;
                    Select  @LastUpdated = GetDate();

--list the tables that are to be pulled back from each DB - if they are not found
the script will not be run against that db

                Declare @ListOfTables Varchar(Max) = 'InvMovements';

--create temporary tables to be pulled from different databases, including a column
to id

                Create Table [#Table1]
                    (
                      [Company] Varchar(150)    collate Latin1_General_BIN
                    , [Bin] Varchar(150)        collate Latin1_General_BIN
                    );

--create script to pull data from each db into the tables

                Declare @Company Varchar(30) = 'All';
                Declare @SQL Varchar(Max) = 'USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end'
                    + --Only query DBs beginning SysProCompany

                    '
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN'
                    + --only companies selected in main run, or if companies
selected then all

                    '
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
                    + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT'
                    + --count number of tables requested (number of commas plus one)

                    '
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')'
                    + --Count of the tables requested how many exist in the db

                    '
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'','')) '
                    + --only if the count matches (all the tables exist in the
requested db) then run the script
```

Author: Johnson, Chris

```
                    '
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
                Insert #Table1
                    ( Company, Bin)
                Select distinct @DBCode
                ,Bin
                From InvMovements
            End
    End';

--Enable this function to check script changes (try to run script directly against
db manually)

--Print @SQL


--execute script against each db, populating the base tables

                Exec [Process].[ExecForEachDB] @cmd = @SQL;

                Insert  [Lookups].[Bin]
                        ( [Company]
                        , [Bin]
                        , [LastUpdated]
                        )
                        Select  [Company]
                              , [Bin]
                              , @LastUpdated
                        From    [#Table1];

                If @PrevCheck = 1
                    Begin
                        Declare @CurrentCount Int
                          , @PreviousCount Int;

                        Select  @CurrentCount = Count(*)
                        From    [Lookups].[Bin]
                        Where   [LastUpdated] = @LastUpdated;

                        Select  @PreviousCount = Count(*)
                        From    [Lookups].[Bin]
                        Where   [LastUpdated] <> @LastUpdated;

                        If @PreviousCount > @CurrentCount
                            Begin
                                Delete  [Lookups].[Bin]
                                Where   [LastUpdated] = @LastUpdated;
                                Print 'UspUpdate_Bin - Count has gone down since
last run, no update applied';
                                Print 'Current Count = '
                                    + Cast(@CurrentCount As Varchar(5))
                                    + ' Previous Count = '
                                    + Cast(@PreviousCount As Varchar(5));
                            End;
                        If @PreviousCount <= @CurrentCount
                            Begin
```

Author: Johnson, Chris

```
                                Delete  [Lookups].[Bin]
                                Where   [LastUpdated] <> @LastUpdated;
                                Print 'UspUpdate_Bin - Update applied successfully';
                            End;
                    End;
                If @PrevCheck = 0
                    Begin
                        Delete  [Lookups].[Bin]
                        Where   [LastUpdated] <> @LastUpdated;
                        Print 'UspUpdate_Bin - Update applied successfully';
                    End;
            End;
    End;
    If DateDiff(Minute , @LastDate , GetDate()) <= (@HoursBetweenUpdates*60)
        Begin
            Print 'UspUpdate_Bin - Table was last updated at '
                + Cast(@LastDate As Varchar(255)) + ' no update applied';
        End;

GO
EXEC sp_addextendedproperty N'MS_Description', N'Stored proc to update specified
table', 'SCHEMA', N'Process', 'PROCEDURE', N'UspUpdate_Bin', NULL, NULL
GO
```

Uses

[Lookups].[Bin]
[Process].[ExecForEachDB]
[Process]

Author: Johnson, Chris

[Process].[UspUpdate_BudgetType]

MS_Description

Stored proc to update specified table

Parameters

Name	Data Type	Max Length (Bytes)
@PrevCheck	int	4
@HoursBetweenUpdates	numeric(5,2)	5

SQL Script

```
CREATE Proc [Process].[UspUpdate_BudgetType]
    (
      @PrevCheck Int
    , @HoursBetweenUpdates Numeric(5 , 2)
    )
As
    Begin
--remove nocount on to speed up query

        Set NoCount On;

--check if table exists and create if it doesn't

        If ( Not Exists ( Select    1
                          From      [INFORMATION_SCHEMA].[TABLES]
                          Where     [TABLE_SCHEMA] = 'Lookups'
                                    And [TABLE_NAME] = 'BudgetType' )
           )
            Begin
                Create Table [Lookups].[BudgetType]
                    (
                      [BudgetType] Char(1)            collate latin1_general_bin
                    , [BudgetTypeDesc] Varchar(250)    collate latin1_general_bin
                    , [LastUpdated] DateTime2
                    );
            End;


        Declare @LastUpdate DateTime2 = GetDate();
        Declare @LastDate DateTime2;

        Select  @LastDate = Max([bt].[LastUpdated])
        From    [Lookups].[BudgetType] As [bt];

        If DateDiff(Minute , @LastDate , GetDate()) > ( @HoursBetweenUpdates
                                                        * 60 )
            Begin
                Insert  [Lookups].[BudgetType]
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Process.UspUpdate_-BudgetType

```
                        ( [BudgetType]
                        , [BudgetTypeDesc]
                        , [LastUpdated]
                        )
                        Select  [BudgetType] = 'C'
                              , [BudgetTypeDesc] = 'Current Year'
                              , [LastUpdated] = @LastUpdate;
                Insert  [Lookups].[BudgetType]
                        ( [BudgetType]
                        , [BudgetTypeDesc]
                        , [LastUpdated]
                        )
                        Select  [BudgetType] = 'N'
                              , [BudgetTypeDesc] = 'Next Year'
                              , [LastUpdated] = @LastUpdate;
                Insert  [Lookups].[BudgetType]
                        ( [BudgetType]
                        , [BudgetTypeDesc]
                        , [LastUpdated]
                        )
                        Select  [BudgetType] = 'A'
                              , [BudgetTypeDesc] = 'Alternate Budget'
                              , [LastUpdated] = @LastUpdate;

                If @PrevCheck = 1
                    Begin
                        Declare @CurrentCount Int
                          , @PreviousCount Int;

                        Select  @CurrentCount = Count(*)
                        From    [Lookups].[BudgetType]
                        Where   [LastUpdated] = @LastUpdate;

                        Select  @PreviousCount = Count(*)
                        From    [Lookups].[BudgetType]
                        Where   [LastUpdated] <> @LastUpdate;

                        If @PreviousCount > @CurrentCount
                            Begin
                                Delete  [Lookups].[BudgetType]
                                Where   [LastUpdated] = @LastUpdate;
                                Print 'UspUpdate_BudgetType - Count has gone down
since last run, no update applied';
                                Print 'Current Count = '
                                    + Cast(@CurrentCount As Varchar(5))
                                    + ' Previous Count = '
                                    + Cast(@PreviousCount As Varchar(5));
                            End;
                        If @PreviousCount <= @CurrentCount
                            Begin
                                Delete  [Lookups].[BudgetType]
                                Where   [LastUpdated] <> @LastUpdate;
                                Print 'UspUpdate_BudgetType - Update applied
successfully';
                            End;
                    End;
                If @PrevCheck = 0
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Process.UspUpdate_-BudgetType

```
                    Begin
                        Delete  [Lookups].[BudgetType]
                        Where   [LastUpdated] <> @LastUpdate;
                        Print 'UspUpdate_BudgetType - Update applied successfully';
                    End;
            End;
        If DateDiff(Minute , @LastDate , GetDate()) <= ( @HoursBetweenUpdates
                                                        * 60 )
            Begin
                Print 'UspUpdate_BudgetType - Table was last updated at '
                    + Cast(@LastDate As Varchar(255)) + ' no update applied';
            End;
    End;
GO
EXEC sp_addextendedproperty N'MS_Description', N'Stored proc to update specified
table', 'SCHEMA', N'Process', 'PROCEDURE', N'UspUpdate_BudgetType', NULL, NULL
GO
```

Uses

[Lookups].[BudgetType]
[Process]

Author: Johnson, Chris

[Process].[UspUpdate_Buyers]

MS_Description

Stored proc to update specified table

Parameters

Name	Data Type	Max Length (Bytes)
@PrevCheck	int	4
@HoursBetweenUpdates	numeric(5,2)	5

SQL Script

```
CREATE Proc [Process].[UspUpdate_Buyers]
    (
      @PrevCheck Int
    , @HoursBetweenUpdates Numeric(5 , 2)
    )
As
    Begin

--check if table exists and create if it doesn't

        If ( Not Exists ( Select    *
                          From      [INFORMATION_SCHEMA].[TABLES]
                          Where     [TABLE_SCHEMA] = 'Lookups'
                                    And [TABLE_NAME] = 'Buyers' )
           )
            Begin
                Create Table [Lookups].[Buyers]
                    (
                      [Company] Varchar(150)    collate Latin1_General_BIN
                    , [BuyerName] Varchar(150)    collate Latin1_General_BIN
                    , [LastUpdated] DateTime2
                    );
            End;



--check last time run and update if it's been longer than @HoursBetweenUpdates hours

        Declare @LastDate DateTime2;

        Select  @LastDate = Max([LastUpdated])
        From    [Lookups].[Buyers];

        If @LastDate Is Null
            Or DateDiff(Minute , @LastDate , GetDate()) > ( @HoursBetweenUpdates
                                                            * 60 )
            Begin
    --Set time of run
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Process.UspUpdate_-Buyers

```
                Declare @LastUpdated DateTime2;
                Select  @LastUpdated = GetDate();

--list the tables that are to be pulled back from each DB - if they are not found
the script will not be run against that db

                Declare @ListOfTables Varchar(Max) = 'InvBuyer';

--create temporary tables to be pulled from different databases, including a column
to id

                Create Table [#Table1]
                    (
                      [Company] Varchar(150)    collate Latin1_General_BIN
                    , [BuyerName] Varchar(150)    collate Latin1_General_BIN
                    );

--create script to pull data from each db into the tables

                Declare @Company Varchar(30) = 'All';
                Declare @SQL Varchar(Max) = 'USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
                    + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables
                    + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_Bin From Black-
Box.dbo.udf_SplitString(@ListOfTables,'',''))
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
                Insert #Table1
                    ( Company, [BuyerName])
                Select distinct @DBCode
                ,Name
                From [InvBuyer]
            End
    End';

--execute script against each db, populating the base tables

                Exec [Process].[ExecForEachDB] @cmd = @SQL;

                Insert  [Lookups].[Buyers]
                        ( [Company]
                        , [BuyerName]
                        , [LastUpdated]
                        )
                        Select  [Company]
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Process.UspUpdate_-Buyers

```
                                , [BuyerName]
                                , @LastUpdated
                        From    [#Table1];

                If @PrevCheck = 1
                    Begin
                        Declare @CurrentCount Int
                          , @PreviousCount Int;

                        Select  @CurrentCount = Count(*)
                        From    [Lookups].[Buyers]
                        Where   [LastUpdated] = @LastUpdated;

                        Select  @PreviousCount = Count(*)
                        From    [Lookups].[Buyers]
                        Where   [LastUpdated] <> @LastUpdated;

                        If @PreviousCount > @CurrentCount
                            Begin
                                Delete  [Lookups].[Buyers]
                                Where   [LastUpdated] = @LastUpdated;
                                Print 'UspUpdate_Buyers - Count has gone down since
last run, no update applied';
                                Print 'Current Count = '
                                    + Cast(@CurrentCount As Varchar(5))
                                    + ' Previous Count = '
                                    + Cast(@PreviousCount As Varchar(5));
                            End;
                        If @PreviousCount <= @CurrentCount
                            Begin
                                Delete  [Lookups].[Buyers]
                                Where   [LastUpdated] <> @LastUpdated;
                                Print 'UspUpdate_Buyers - Update applied
successfully';
                            End;
                    End;
                If @PrevCheck = 0
                    Begin
                        Delete  [Lookups].[Buyers]
                        Where   [LastUpdated] <> @LastUpdated;
                        Print 'UspUpdate_Buyers - Update applied successfully';
                    End;
            End;
    End;
    If DateDiff(Minute , @LastDate , GetDate()) <= ( @HoursBetweenUpdates * 60 )
        Begin
            Print 'UspUpdate_Buyers - Table was last updated at '
                + Cast(@LastDate As Varchar(255)) + ' no update applied';
        End;

GO
EXEC sp_addextendedproperty N'MS_Description', N'Stored proc to update specified
table', 'SCHEMA', N'Process', 'PROCEDURE', N'UspUpdate_Buyers', NULL, NULL
GO
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Process.UspUpdate_-Buyers

Uses

[Lookups].[Buyers]

[Process].[ExecForEachDB]

[Process]

Author: Johnson, Chris

[Process].[UspUpdate_CompanyNames]

MS_Description

Stored proc to update specified table

Parameters

Name	Data Type	Max Length (Bytes)
@PrevCheck	int	4
@HoursBetweenUpdates	numeric(5,2)	5

SQL Script

```
CREATE Proc [Process].[UspUpdate_CompanyNames]
    (
      @PrevCheck Int --if count is less than previous don't update

    , @HoursBetweenUpdates Numeric(5 , 2)
    )
As
    Begin

--check if table exists with latest field to drop if it doesn't

        If ( Not Exists ( Select    1
                          From      [sys].[tables] As [T]
                                    Left Join [sys].[schemas] As [S]
                                        On [S].[schema_id] = [T].[schema_id]
                                    Left Join [sys].[columns] As [C]
                                        On [C].[object_id] = [T].[object_id]
                          Where     [S].[name] = 'Lookups'
                                    And [T].[name] = 'CompanyNames'
                                    And [C].[name] = 'ShortName' )
           )
            Begin
                Begin Try
                    Drop Table [Lookups].[CompanyNames];
                End Try
                Begin Catch
                    Print 'unable to drop table - may not exist';
                End Catch;
            End;
--check if table exists to create it

        If ( Not Exists ( Select    1
                          From      [sys].[tables] As [T]
                                    Left Join [sys].[schemas] As [S]
                                        On [S].[schema_id] = [T].[schema_id]
                          Where     [S].[name] = 'Lookups'
                                    And [T].[name] = 'CompanyNames' )
           )
```

Author: Johnson, Chris

```
            Begin
                Create Table [Lookups].[CompanyNames]
                    (
                      [Company] Varchar(150)        collate Latin1_General_BIN
                    , [CompanyName] Varchar(250)    collate Latin1_General_BIN
                    , [ShortName] Varchar(250)        collate Latin1_General_BIN
                    , [Currency] Varchar(10)        collate Latin1_General_BIN
                    , [LastUpdated] DateTime2
                    );
            End;

--check last time run and update if it's been longer than @HoursBetweenUpdates hours

        Declare @LastDate DateTime2;

        Select  @LastDate = Max([CN].[LastUpdated])
        From    [Lookups].[CompanyNames] As [CN];

        If @LastDate Is Null
            Or DateDiff(Minute , @LastDate , GetDate()) > ( @HoursBetweenUpdates
                                                            * 60 )
            Begin
    --Set time of run

                Declare @LastUpdated DateTime2;
                Select  @LastUpdated = GetDate();

    --create master list of Company Names

                Create     Table [#CompanyList]
                    (
                      [Company] Varchar(150)        collate Latin1_General_BIN
                    , [CompanyName] Varchar(250)    collate Latin1_General_BIN
                    , [ShortName] Varchar(250)        collate Latin1_General_BIN
                    );


                Insert  [#CompanyList]
                        ( [Company]
                        , [CompanyName]
                        , [ShortName]
                        )
                        Select  [t].[Company]
                              , [t].[CompanyName]
                              , [t].[ShortName]
                        From    ( Select    [Company] = '0'
                                          , [CompanyName] = 'Prometic BioSciences
Ltd. TEST'
                                          , [ShortName] = 'N/A'
                                  Union
                                  Select    [Company] = '10'
                                          , [CompanyName] = 'Prometic BioSciences
Ltd'
                                          , [ShortName] = 'PBL'
                                  Union
                                  Select    [Company] = '11'
                                          , [CompanyName] = 'Prometic
Biotherapeutics Ltd'
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Process.UspUpdate_-CompanyNames

```
                                        , [ShortName] = 'PBT Ltd'
                                Union
                                Select    [Company] = '12'
                                        , [CompanyName] = 'Prometic Pharma SMT
Ltd'
                                        , [ShortName] = 'PSMT'
                                Union
                                Select    [Company] = '13'
                                        , [CompanyName] = 'Prometic Pharma SMTH
Ltd'
                                        , [ShortName] = 'PSMH'
                                Union
                                Select    [Company] = '20'
                                        , [CompanyName] = 'Prometic
Biotherapeutics Inc'
                                        , [ShortName] = 'PBT'
                                Union
                                Select    [Company] = '21'
                                        , [CompanyName] = 'Pathogen Removal Device
Tech'
                                        , [ShortName] = 'PRDT'
                                Union
                                Select    [Company] = '22'
                                        , [CompanyName] = 'Nantpro'
                                        , [ShortName] = 'Nantpro'
                                Union
                                Select    [Company] = '40'
                                        , [CompanyName] = 'Prometic Life Sciences
Inc'
                                        , [ShortName] = 'PLI'
                                Union
                                Select    [Company] = '41'
                                        , [CompanyName] = 'Prometic Biosciences
Inc'
                                        , [ShortName] = 'PBI'
                                Union
                                Select    [Company] = '42'
                                        , [CompanyName] = 'Prometic Manufacturing
Inc'
                                        , [ShortName] = 'PMI'
                                Union
                                Select    [Company] = '43'
                                        , [CompanyName] = 'Prometic Bioproduction
Inc'
                                        , [ShortName] = 'PBP'
                                Union
                                Select    [Company] = '44'
                                        , [CompanyName] = 'Prometic Plasma
Resources Inc'
                                        , [ShortName] = 'PPR'
                                Union
                                Select    [Company] = '70'
                                        , [CompanyName] = 'Prometic Biosciences
Russia'
                                        , [ShortName] = 'PBR Russia'
                                Union
                                Select    [Company] = '91'
                                        , [CompanyName] = 'Prometic Biosciences
Inc - Dec Y/E'
                                        , [ShortName] = 'PBI - Dec Y/E'
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Process.UspUpdate_-CompanyNames

```
                                    Union
                                    Select    [Company] = '92'
                                            , [CompanyName] = 'Prometic
Biotherapeutics Inc'
                                            , [ShortName] = 'PBT Inc'
                                    Union
                                    Select    [Company] = 'C'
                                            , [CompanyName] = 'Prometic
Biotherapeutics TEST'
                                            , [ShortName] = 'N/A'
                                    Union
                                    Select    [Company] = 'D'
                                            , [CompanyName] = 'Prometic Biosciences
Inc TEST'
                                            , [ShortName] = 'N/A'
                                    Union
                                    Select    [Company] = 'F'
                                            , [CompanyName] = 'Prometic Biosciences
Inc TEST'
                                            , [ShortName] = 'N/A'
                                    Union
                                    Select    [Company] = 'G'
                                            , [CompanyName] = 'Pathogen Removal Device
Tech TEST'
                                            , [ShortName] = 'N/A'
                                    Union
                                    Select    [Company] = 'H'
                                            , [CompanyName] = 'Prometic
Biotherapeutics TEST'
                                            , [ShortName] = 'N/A'
                                    Union
                                    Select    [Company] = 'P'
                                            , [CompanyName] = 'Prometic Bioproduction
TEST'
                                            , [ShortName] = 'N/A'
                                    Union
                                    Select    [Company] = 'Q'
                                            , [CompanyName] = 'Prometic Life Sciences
TEST'
                                            , [ShortName] = 'N/A'
                                    Union
                                    Select    [Company] = 'T'
                                            , [CompanyName] = 'Prometic Biosciences
Ltd TEST'
                                            , [ShortName] = 'N/A'
                                    Union
                                    Select    [Company] = 'U'
                                            , [CompanyName] = 'Prometic Manufacturing
TEST'
                                            , [ShortName] = 'N/A'
                                    Union
                                    Select    [Company] = 'V'
                                            , [CompanyName] = 'Pathogen Removal Device
TEST'
                                            , [ShortName] = 'N/A'
                                ) [t];

    --Get list of all companies in use

                Create Table [#CompanyNameTable1]
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Process.UspUpdate_-CompanyNames

```
                    (
                      [Company] Varchar(150) Collate Latin1_General_BIN
                    , [Currency] Varchar(10) Collate Latin1_General_BIN
                    );

    --create script to pull data from each db into the tables

                Declare @SQL Varchar(Max) = 'USE [?];
            Declare @DB varchar(150),@DBCode varchar(150)
            Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end
            IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
            BEGIN
                Insert #CompanyNameTable1
                    (Company, Currency)
                Select
                    @DBCode,[TC].[Currency]
                FROM [dbo].[TblCurrency] As [TC]
            Where [TC].[BuyExchangeRate]=1
            End';

    --execute script against each db, populating the base tables

                Exec [Process].[ExecForEachDB] @cmd = @SQL;

                Select  [T].[Company]
                      , [CompanyName] = Coalesce([cl].[CompanyName] ,
                                                 'Unknown')
                      , [ShortName] = Coalesce([cl].[ShortName] , 'Unknown')
                      , [T].[Currency]
                Into    [#ResultsCompanyName]
                From    [#CompanyNameTable1] [T]
                        Left Join [#CompanyList] As [cl]
                            On [cl].[Company] = [T].[Company];


    --placeholder for anomalous results that are different to master list

                Insert  [Lookups].[CompanyNames]
                        ( [Company]
                        , [CompanyName]
                        , [Currency]
                        , [ShortName]
                        , [LastUpdated]
                        )
                        Select  [rcn].[Company]
                              , [rcn].[CompanyName]
                              , [rcn].[Currency]
                              , [rcn].[ShortName]
                              , @LastUpdated
                        From    [#ResultsCompanyName] As [rcn];

                If @PrevCheck = 1
                    Begin
                        Declare @CurrentCount Int
                          , @PreviousCount Int;
```

Author: Johnson, Chris

```
                        Select  @CurrentCount = Count(*)
                        From    [Lookups].[CompanyNames] As [cn]
                        Where   [cn].[LastUpdated] = @LastUpdated;

                        Select  @PreviousCount = Count(*)
                        From    [Lookups].[CompanyNames] As [cn]
                        Where   [cn].[LastUpdated] <> @LastUpdated;



                        If @PreviousCount > @CurrentCount
                            Begin
                                Delete  [Lookups].[CompanyNames]
                                Where   [LastUpdated] = @LastDate;
                                Print 'UspUpdate_CompanyNames - Count has gone down
since last run, no update applied';
                                Print 'Current Count = '
                                    + Cast(@CurrentCount As Varchar(5))
                                    + ' Previous Count = '
                                    + Cast(@PreviousCount As Varchar(5));
                            End;
                        If @PreviousCount <= @CurrentCount
                            Begin
                                Delete  [Lookups].[CompanyNames]
                                Where   [LastUpdated] <> @LastUpdated
                                        Or [LastUpdated] Is Null;
                                Print 'UspUpdate_CompanyNames - Update applied
successfully';
                            End;
                    End;
                If @PrevCheck = 0
                    Begin
                        Delete  [Lookups].[CompanyNames]
                        Where   [LastUpdated] <> @LastUpdated;
                        Print 'UspUpdate_CompanyNames - Update applied
successfully';
                    End;

            End;
        If DateDiff(Minute , @LastDate , GetDate()) <= ( @HoursBetweenUpdates
                                                          * 60 )
            Begin
                Print 'UspUpdate_CompanyNames - Table was last updated at '
                    + Cast(@LastDate As Varchar(255)) + ' no update applied';
            End;
    End;

GO
EXEC sp_addextendedproperty N'MS_Description', N'Stored proc to update specified
table', 'SCHEMA', N'Process', 'PROCEDURE', N'UspUpdate_CompanyNames', NULL, NULL
GO
```

Uses

[Lookups].[CompanyNames]
[Process].[ExecForEachDB]

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Process.UspUpdate_-CompanyNames

[Process]

Author: Johnson, Chris

[Process].[UspUpdate_CurrencyRates]

MS_Description

Stored proc to update specified table

Parameters

Name	Data Type	Max Length (Bytes)
@PrevCheck	int	4
@HoursBetweenUpdates	numeric(5,2)	5

SQL Script

```
CREATE Proc [Process].[UspUpdate_CurrencyRates]
    (
      @PrevCheck Int --if count is less than previous don't update

    , @HoursBetweenUpdates Numeric(5,2)
    )
As
    Begin
Set NoCount on
        Declare @MaxRank Int;

--check if table exists and create if it doesn't

        If ( Not Exists ( Select    1
                          From      [INFORMATION_SCHEMA].[TABLES]
                          Where     [TABLE_SCHEMA] = 'Lookups'
                                    And [TABLE_NAME] = 'CurrencyRates' )
           )
            Begin
                Create Table [Lookups].[CurrencyRates]
                    (
                      [StartDateTime] DateTime
                    , [EndDateTime] DateTime
                    , [Currency] Varchar(10)            collate latin1_general_bin
                    , [CADDivision] Numeric(12 , 7)
                    , [CHFDivision] Numeric(12 , 7)
                    , [EURDivision] Numeric(12 , 7)
                    , [GBPDivision] Numeric(12 , 7)
                    , [JPYDivision] Numeric(12 , 7)
                    , [USDDivision] Numeric(12 , 7)
                    , [CADMultiply] Numeric(12 , 7)
                    , [CHFMultiply] Numeric(12 , 7)
                    , [EURMultiply] Numeric(12 , 7)
                    , [GBPMultiply] Numeric(12 , 7)
                    , [JPYMultiply] Numeric(12 , 7)
                    , [USDMultiply] Numeric(12 , 7)
                    , [LastUpdated] DateTime2
                    );
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Process.UspUpdate_-CurrencyRates

```
            End;

--check last time run and update if it's been longer than @HoursBetweenUpdates hours

        Declare @LastDate DateTime2;

        Select  @LastDate = Max([CR].[LastUpdated])
        From    [Lookups].[CurrencyRates] As [CR];

        If @LastDate Is Null
            Or DateDiff(Minute , @LastDate , GetDate()) > (@HoursBetweenUpdates*60)
            Begin
--Set time of run

                Declare @LastUpdated DateTime2;
                Select  @LastUpdated = GetDate();

--Create table to capture currencies

                Create Table [#MasterCurrencyTable]
                    (
                      [JnlDate] Date
                    , [StartDateTime] DateTime
                    , [EndDateTime] DateTime
                    , [USD] Float
                    , [JPY] Float
                    , [EUR] Float
                    , [GBP] Float
                    , [CAD] Float
                    , [CHF] Float
                    , [JnlRank] Int
                    );


--Grab

                Insert  [#MasterCurrencyTable]
                        ( [JnlDate]
                        , [StartDateTime]
                        , [EndDateTime]
                        , [USD]
                        , [JPY]
                        , [EUR]
                        , [GBP]
                        , [CAD]
                        )
                        Select  [JnlDate]
                              , [StartDateTime] = [JnlDateTime]
                              , [EndDateTime] = Null
                              , [USD]
                              , [JPY]
                              , [EUR]
                              , [GBP]
                              , [CAD] = 1.000000
                        From    ( Select    [TCAJ].[JnlDate]
                                          , [JnlDateTime] = DateAdd(Hour ,
                                                                    Cast(Left([TCAJ].[Jnl-
```

Author: Johnson, Chris

```
Time] ,
                                                              2) As Int) ,
                                                              [TCAJ].[JnlDate])
                                            , [TCAJ].[Currency]
                                            , [TCAJ].[ColumnName]
                                            , [ExchangeRate] = Cast([TCAJ].[After] As
Float)
                                    From    [SysproCompany40].[dbo].[TblCurAmendment-
Jnl]
                                            As [TCAJ]
                                    Where   [TCAJ].[ColumnName] = 'BuyExchangeRate'
                                  ) [t] Pivot ( Max([ExchangeRate]) For [Currency] In
( [USD] ,
                                                              [JPY] , [EUR] ,
                                                              [GBP] ) ) [t];

        --Get Swiss francs from co. 10

                Insert  [#MasterCurrencyTable]
                        ( [JnlDate]
                        , [StartDateTime]
                        , [EndDateTime]
                        , [USD]
                        , [JPY]
                        , [EUR]
                        , [GBP]
                        , [CAD]
                        , [CHF]
                        )
                        Select  [JnlDateTime]
                              , [StartDateTime] = [JnlDateTime]
                              , [EndDateTime] = Null
                              , [USD] = [CAD] / [USD]
                              , [JPY] = [CAD] / [JPY]
                              , [EUR] = [CAD] / [EUR]
                              , [GBP] = [CAD]
                              , [CAD] = [CAD] / [CAD]
                              , [CHF] = [CAD] / [CHF]
                        From    ( Select    [JnlDateTime] = [TCAJ].[JnlDate]
                                          , [TCAJ].[Currency]
                                          , [TCAJ].[ColumnName]
                                          , [ExchangeRate] = Cast([TCAJ].[After] As
Float)
                                    From    [SysproCompany10].[dbo].[TblCurAmendment-
Jnl]
                                            As [TCAJ]
                                    Where   [TCAJ].[ColumnName] = 'BuyExchangeRate'
                                  ) [t] Pivot ( Max([ExchangeRate]) For [Currency] In
( [USD] ,
                                                              [JPY] , [EUR] ,
                                                              [GBP] , [CHF] ,
                                                              [CAD] ) ) [t]
                        Where   [CHF] Is Not Null
                                And [CAD] / [CAD] Is Not Null
                        Order By [t].[JnlDateTime] Asc;

--Add rank
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Process.UspUpdate_-CurrencyRates

```
                Update  [#MasterCurrencyTable]
                Set     [JnlRank] = [MCTb].[Ranking]
                From    [#MasterCurrencyTable] As [MCT]
                        Inner Join ( Select [MCTa].[StartDateTime]
                                          , [Ranking] = Rank() Over ( Order By
[MCTa].[StartDateTime] )
                                     From   [#MasterCurrencyTable] As [MCTa]
                                   ) As [MCTb]
                            On [MCTb].[StartDateTime] = [MCT].[StartDateTime];

 --Set End Dates

                Update  [MCT]
                Set     [MCT].[EndDateTime] = DateAdd(Second , -1 ,
                                                      [MCT2].[StartDateTime])
                From    [#MasterCurrencyTable] As [MCT]
                        Left Join [#MasterCurrencyTable] As [MCT2]
                            On [MCT2].[JnlRank] = [MCT].[JnlRank] + 1;

 --Back fill null currencies by grabbing the next available rate

                Update  [#MasterCurrencyTable]
                Set     [JPY] = ( Select Top 1
                                         [MCT2].[JPY]
                                  From      [#MasterCurrencyTable] As [MCT2]
                                  Where     [MCT2].[StartDateTime] > [#Master-
CurrencyTable].[StartDateTime]
                                            And [MCT2].[JPY] Is Not Null
                                  Order By  [MCT2].[StartDateTime] Asc
                                )
                Where   [#MasterCurrencyTable].[JPY] Is Null;
                Update  [#MasterCurrencyTable]
                Set     [USD] = ( Select Top 1
                                         [MCT2].[USD]
                                  From      [#MasterCurrencyTable] As [MCT2]
                                  Where     [MCT2].[StartDateTime] > [#Master-
CurrencyTable].[StartDateTime]
                                            And [MCT2].[USD] Is Not Null
                                  Order By  [MCT2].[StartDateTime] Asc
                                )
                Where   [#MasterCurrencyTable].[USD] Is Null;
                Update  [#MasterCurrencyTable]
                Set     [EUR] = ( Select Top 1
                                         [MCT2].[EUR]
                                  From      [#MasterCurrencyTable] As [MCT2]
                                  Where     [MCT2].[StartDateTime] > [#Master-
CurrencyTable].[StartDateTime]
                                            And [MCT2].[EUR] Is Not Null
                                  Order By  [MCT2].[StartDateTime] Asc
                                )
                Where   [#MasterCurrencyTable].[EUR] Is Null;
                Update  [#MasterCurrencyTable]
                Set     [GBP] = ( Select Top 1
                                         [MCT2].[GBP]
                                  From      [#MasterCurrencyTable] As [MCT2]
                                  Where     [MCT2].[StartDateTime] > [#Master-
CurrencyTable].[StartDateTime]
                                            And [MCT2].[GBP] Is Not Null
```

Author: Johnson, Chris

```
                                    Order By  [MCT2].[StartDateTime] Asc
                                  )
                Where   [#MasterCurrencyTable].[GBP] Is Null;
                Update  [#MasterCurrencyTable]
                Set     [CHF] = ( Select Top 1
                                          [MCT2].[CHF]
                                    From      [#MasterCurrencyTable] As [MCT2]
                                    Where     [MCT2].[StartDateTime] > [#Master-
CurrencyTable].[StartDateTime]
                                          And [MCT2].[CHF] Is Not Null
                                    Order By  [MCT2].[StartDateTime] Asc
                                  )
                Where   [#MasterCurrencyTable].[CHF] Is Null;
--forward fill null currency by grabbing the last available rate

                Update  [#MasterCurrencyTable]
                Set     [JPY] = ( Select Top 1
                                          [MCT2].[JPY]
                                    From      [#MasterCurrencyTable] As [MCT2]
                                    Where     [MCT2].[StartDateTime] < [#Master-
CurrencyTable].[StartDateTime]
                                          And [MCT2].[JPY] Is Not Null
                                    Order By  [MCT2].[StartDateTime] Asc
                                  )
                Where   [#MasterCurrencyTable].[JPY] Is Null;
                Update  [#MasterCurrencyTable]
                Set     [USD] = ( Select Top 1
                                          [MCT2].[USD]
                                    From      [#MasterCurrencyTable] As [MCT2]
                                    Where     [MCT2].[StartDateTime] < [#Master-
CurrencyTable].[StartDateTime]
                                          And [MCT2].[USD] Is Not Null
                                    Order By  [MCT2].[StartDateTime] Asc
                                  )
                Where   [#MasterCurrencyTable].[USD] Is Null;
                Update  [#MasterCurrencyTable]
                Set     [EUR] = ( Select Top 1
                                          [MCT2].[EUR]
                                    From      [#MasterCurrencyTable] As [MCT2]
                                    Where     [MCT2].[StartDateTime] < [#Master-
CurrencyTable].[StartDateTime]
                                          And [MCT2].[EUR] Is Not Null
                                    Order By  [MCT2].[StartDateTime] Asc
                                  )
                Where   [#MasterCurrencyTable].[EUR] Is Null;
                Update  [#MasterCurrencyTable]
                Set     [GBP] = ( Select Top 1
                                          [MCT2].[GBP]
                                    From      [#MasterCurrencyTable] As [MCT2]
                                    Where     [MCT2].[StartDateTime] < [#Master-
CurrencyTable].[StartDateTime]
                                          And [MCT2].[GBP] Is Not Null
                                    Order By  [MCT2].[StartDateTime] Asc
                                  )
                Where   [#MasterCurrencyTable].[GBP] Is Null;
                Update  [#MasterCurrencyTable]
                Set     [CHF] = ( Select Top 1
                                          [MCT2].[CHF]
```

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Process.UspUpdate_-CurrencyRates

```
                                    From      [#MasterCurrencyTable] As [MCT2]
                                    Where     [MCT2].[StartDateTime] < [#Master-
CurrencyTable].[StartDateTime]
                                              And [MCT2].[CHF] Is Not Null
                                    Order By  [MCT2].[StartDateTime] Asc
                                  )
                Where   [#MasterCurrencyTable].[CHF] Is Null;


--get the latest rank to determine which row doesn't have an end date

                Select  @MaxRank = Max([MCT].[JnlRank])
                From    [#MasterCurrencyTable] As [MCT];

--For the latest row, set the end date to tomorrow

                Update  [#MasterCurrencyTable]
                Set     [EndDateTime] = GetDate() + 1
                Where   [JnlRank] = @MaxRank
                        And [EndDateTime] Is Null;

                Insert  [Lookups].[CurrencyRates]
                        ( [StartDateTime]
                        , [EndDateTime]
                        , [Currency]
                        , [CADDivision]
                        , [EURDivision]
                        , [GBPDivision]
                        , [JPYDivision]
                        , [USDDivision]
                        , [CHFDivision]
                        , [CADMultiply]
                        , [EURMultiply]
                        , [GBPMultiply]
                        , [JPYMultiply]
                        , [USDMultiply]
                        , [CHFMultiply]
                        , [LastUpdated]
                        )
                        Select  [t].[StartDateTime]
                              , [t].[EndDateTime]
                              , [t].[Currency]
                              , [CADDivision] = [t].[CAD]
                              , [EURDivision] = [t].[EUR]
                              , [GBPDivision] = [t].[GBP]
                              , [JPYDivision] = [t].[JPY]
                              , [USDDivision] = [t].[USD]
                              , [CHFDivision] = [t].[CHF]
                              , [CADMultiply] = 1 / [t].[CAD]
                              , [EURMultiply] = 1 / [t].[EUR]
                              , [GBPMultiply] = 1 / [t].[GBP]
                              , [JPYMultiply] = 1 / [t].[JPY]
                              , [USDMultiply] = 1 / [t].[USD]
                              , [CHFMultiply] = 1 / [t].[CHF]
                              , @LastUpdated
                        From    ( Select    [MCT].[JnlDate]
                                          , [MCT].[StartDateTime]
```

Author: Johnson, Chris

```
                                        , [MCT].[EndDateTime]
                                        , [Currency] = 'CAD'
                                        , [MCT].[USD]
                                        , [MCT].[JPY]
                                        , [MCT].[EUR]
                                        , [MCT].[GBP]
                                        , [MCT].[CAD]
                                        , [MCT].[CHF]
                                From    [#MasterCurrencyTable] As [MCT]
                                Union All
                                Select  [MCT].[JnlDate]
                                        , [MCT].[StartDateTime]
                                        , [MCT].[EndDateTime]
                                        , [Currency] = 'USD'
                                        , [MCT].[USD] / [MCT].[USD]
                                        , [MCT].[JPY] / [MCT].[USD]
                                        , [MCT].[EUR] / [MCT].[USD]
                                        , [MCT].[GBP] / [MCT].[USD]
                                        , [MCT].[CAD] / [MCT].[USD]
                                        , [MCT].[CHF] / [MCT].[USD]
                                From    [#MasterCurrencyTable] As [MCT]
                                Union All
                                Select  [MCT].[JnlDate]
                                        , [MCT].[StartDateTime]
                                        , [MCT].[EndDateTime]
                                        , [Currency] = 'JPY'
                                        , [MCT].[USD] / [MCT].[JPY]
                                        , [MCT].[JPY] / [MCT].[JPY]
                                        , [MCT].[EUR] / [MCT].[JPY]
                                        , [MCT].[GBP] / [MCT].[JPY]
                                        , [MCT].[CAD] / [MCT].[JPY]
                                        , [MCT].[CHF] / [MCT].[JPY]
                                From    [#MasterCurrencyTable] As [MCT]
                                Union All
                                Select  [MCT].[JnlDate]
                                        , [MCT].[StartDateTime]
                                        , [MCT].[EndDateTime]
                                        , [Currency] = 'EUR'
                                        , [MCT].[USD] / [MCT].[EUR]
                                        , [MCT].[JPY] / [MCT].[EUR]
                                        , [MCT].[EUR] / [MCT].[EUR]
                                        , [MCT].[GBP] / [MCT].[EUR]
                                        , [MCT].[CAD] / [MCT].[EUR]
                                        , [MCT].[CHF] / [MCT].[EUR]
                                From    [#MasterCurrencyTable] As [MCT]
                                Union All
                                Select  [MCT].[JnlDate]
                                        , [MCT].[StartDateTime]
                                        , [MCT].[EndDateTime]
                                        , [Currency] = 'GBP'
                                        , [MCT].[USD] / [MCT].[GBP]
                                        , [MCT].[JPY] / [MCT].[GBP]
                                        , [MCT].[EUR] / [MCT].[GBP]
                                        , [MCT].[GBP] / [MCT].[GBP]
                                        , [MCT].[CAD] / [MCT].[GBP]
                                        , [MCT].[CHF] / [MCT].[GBP]
                                From    [#MasterCurrencyTable] As [MCT]
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Process.UspUpdate_-CurrencyRates

```
                                    Union All
                                    Select    [MCT].[JnlDate]
                                            , [MCT].[StartDateTime]
                                            , [MCT].[EndDateTime]
                                            , [Currency] = 'CHF'
                                            , [MCT].[USD] / [MCT].[CHF]
                                            , [MCT].[JPY] / [MCT].[CHF]
                                            , [MCT].[EUR] / [MCT].[CHF]
                                            , [MCT].[GBP] / [MCT].[CHF]
                                            , [MCT].[CAD] / [MCT].[CHF]
                                            , [MCT].[CHF] / [MCT].[CHF]
                                    From      [#MasterCurrencyTable] As [MCT]
                                  ) [t]
                        Order By [t].[JnlDate] Asc
                              , [t].[StartDateTime] Asc
                              , [t].[Currency] Asc;

                Drop Table [#MasterCurrencyTable];
                If @PrevCheck = 1
                    Begin
                        Declare @CurrentCount Int
                          , @PreviousCount Int;

                        Select  @CurrentCount = Count(*)
                        From    [Lookups].[CurrencyRates]
                        Where   [LastUpdated] = @LastUpdated;

                        Select  @PreviousCount = Count(*)
                        From    [Lookups].[CurrencyRates]
                        Where   [LastUpdated] <> @LastUpdated;

                        If @PreviousCount > @CurrentCount
                            Begin
                                Delete  [Lookups].[CurrencyRates]
                                Where   [LastUpdated] = @LastUpdated;
                                Print 'UspUpdate_CurrencyRates - Count has gone down
since last run, no update applied';
                                Print 'Current Count = '
                                    + Cast(@CurrentCount As Varchar(5))
                                    + ' Previous Count = '
                                    + Cast(@PreviousCount As Varchar(5));
                            End;
                        If @PreviousCount <= @CurrentCount
                            Begin
                                Delete  [Lookups].[CurrencyRates]
                                Where   [LastUpdated] <> @LastUpdated;
                                Print 'UspUpdate_CurrencyRates - Update applied
successfully';
                            End;
                    End;
                If @PrevCheck = 0
                    Begin
                        Delete  [Lookups].[CurrencyRates]
                        Where   [LastUpdated] <> @LastUpdated;
                        Print 'UspUpdate_CurrencyRates - Update applied
successfully';
                    End;
            End;
```

Author: Johnson, Chris

```
    End;
    If DateDiff(Minute , @LastDate , GetDate()) <= (@HoursBetweenUpdates*60)
        Begin
            Print 'UspUpdate_CurrencyRates - Table was last updated at '
                + Cast(@LastDate As Varchar(255)) + ' no update applied';
        End;
GO
EXEC sp_addextendedproperty N'MS_Description', N'Stored proc to update specified
table', 'SCHEMA', N'Process', 'PROCEDURE', N'UspUpdate_CurrencyRates', NULL, NULL
GO
```

Uses

[Lookups].[CurrencyRates]

[Process]

Author: Johnson, Chris

[Process].[UspUpdate_GenJournalCtlJnlSource]

MS_Description

Stored proc to update specified table

Parameters

Name	Data Type	Max Length (Bytes)
@PrevCheck	int	4
@HoursBetweenUpdates	numeric(5,2)	5

SQL Script

```
CREATE Proc [Process].[UspUpdate_GenJournalCtlJnlSource]
    (
      @PrevCheck Int
    , @HoursBetweenUpdates Numeric(5 , 2)
    )
As
    Begin


--remove nocount on to speed up query

        Set NoCount On;

--check if table exists and create if it doesn't

        If ( Not Exists ( Select    *
                          From      [INFORMATION_SCHEMA].[TABLES]
                          Where     [TABLE_SCHEMA] = 'Lookups'
                                    And [TABLE_NAME] = 'GenJournalCtlJnlSource' )
           )
            Begin
                Create Table [Lookups].[GenJournalCtlJnlSource]
                    (
                      [GenJournalCtlJnlSource] Char(2) Collate Latin1_General_BIN
                    , [GenJournalCtlJnlSourceDesc] Varchar(250) Collate Latin1_-
General_BIN
                    , [LastUpdated] DateTime2
                    );
            End;


        Declare @LastUpdate DateTime2 = GetDate();
        Declare @LastDate DateTime2;

        Select  @LastDate = Max([bt].[LastUpdated])
        From    [Lookups].[GenJournalCtlJnlSource] As [bt];
        If DateDiff(Minute , @LastDate , GetDate()) > ( @HoursBetweenUpdates
                                                        * 60 )
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Process.UspUpdate_-GenJournalCtlJnlSource

```
            Begin
                Insert  [Lookups].[GenJournalCtlJnlSource]
                        ( [GenJournalCtlJnlSource]
                        , [GenJournalCtlJnlSourceDesc]
                        , [LastUpdated]
                        )
                        Select  [t].[GenJournalCtlJnlSource]
                              , [t].[GenJournalCtlJnlSourceDesc]
                              , [LastUpdated] = @LastUpdate
                        From    ( Select    [GenJournalCtlJnlSource] = 'AP'
                                          , [GenJournalCtlJnlSourceDesc] = 'Accounts
Payable'
                                  Union
                                  Select    [GenJournalCtlJnlSource] = 'AR'
                                          , [GenJournalCtlJnlSourceDesc] = 'Accounts
Receivable'
                                  Union
                                  Select    [GenJournalCtlJnlSource] = 'AS'
                                          , [GenJournalCtlJnlSourceDesc] =
'Assets/Assets Register'
                                  Union
                                  Select    [GenJournalCtlJnlSource] = 'GR'
                                          , [GenJournalCtlJnlSourceDesc] = 'Grn/Grn
Matching'
                                  Union
                                  Select    [GenJournalCtlJnlSource] = 'IN'
                                          , [GenJournalCtlJnlSourceDesc] =
'Inventory'
                                  Union
                                  Select    [GenJournalCtlJnlSource] = 'SA'
                                          , [GenJournalCtlJnlSourceDesc] = 'Sales'
                                  Union
                                  Select    [GenJournalCtlJnlSource] = 'WP'
                                          , [GenJournalCtlJnlSourceDesc] = 'Work in
Progress'
                                  Union
                                  Select    [GenJournalCtlJnlSource] = 'PA'
                                          , [GenJournalCtlJnlSourceDesc] = 'Payroll'
                                  Union
                                  Select    [GenJournalCtlJnlSource] = 'CS'
                                          , [GenJournalCtlJnlSourceDesc] =
'Cashbook'
                                ) [t];


                If @PrevCheck = 1
                    Begin
                        Declare @CurrentCount Int
                          , @PreviousCount Int;

                        Select  @CurrentCount = Count(*)
                        From    [Lookups].[GenJournalCtlJnlSource]
                        Where   [LastUpdated] = @LastUpdate;

                        Select  @PreviousCount = Count(*)
                        From    [Lookups].[GenJournalCtlJnlSource]
                        Where   [LastUpdated] <> @LastUpdate;

                        If @PreviousCount > @CurrentCount
```

Author: Johnson, Chris

```
                            Begin
                                Delete  [Lookups].[GenJournalCtlJnlSource]
                                Where   [LastUpdated] = @LastUpdate;
                                Print 'UspUpdate_GenJournalCtlJnlSource - Count has
gone down since last run, no update applied';
                                Print 'Current Count = '
                                    + Cast(@CurrentCount As Varchar(5))
                                    + ' Previous Count = '
                                    + Cast(@PreviousCount As Varchar(5));
                            End;
                        If @PreviousCount <= @CurrentCount
                            Begin
                                Delete  [Lookups].[GenJournalCtlJnlSource]
                                Where   [LastUpdated] <> @LastUpdate;
                                Print 'UspUpdate_GenJournalCtlJnlSource - Update
applied successfully';
                            End;
                    End;
                If @PrevCheck = 0
                    Begin
                        Delete  [Lookups].[GenJournalCtlJnlSource]
                        Where   [LastUpdated] <> @LastUpdate;
                        Print 'UspUpdate_GenJournalCtlJnlSource - Update applied
successfully';
                    End;
            End;
        If DateDiff(Minute , @LastDate , GetDate()) <= ( @HoursBetweenUpdates
                                                          * 60 )
            Begin
                Print 'UspUpdate_GenJournalCtlJnlSource - Table was last updated at
'
                    + Cast(@LastDate As Varchar(255)) + ' no update applied';
            End;
    End;
GO
EXEC sp_addextendedproperty N'MS_Description', N'Stored proc to update specified
table', 'SCHEMA', N'Process', 'PROCEDURE', N'UspUpdate_GenJournalCtlJnlSource',
NULL, NULL
GO
```

Uses

[Lookups].[GenJournalCtlJnlSource]
[Process]

Author: Johnson, Chris

[Process].[UspUpdate_GenTransactionSource]

MS_Description

Stored proc to update specified table

Parameters

Name	Data Type	Max Length (Bytes)
@PrevCheck	int	4
@HoursBetweenUpdates	numeric(5,2)	5

SQL Script

```
CREATE Proc [Process].[UspUpdate_GenTransactionSource]
    (
      @PrevCheck Int
    , @HoursBetweenUpdates Numeric(5 , 2)
    )
As
    Begin
--remove nocount on to speed up query

        Set NoCount On;

--check if table exists and create if it doesn't

        If ( Not Exists ( Select    1
                          From      [INFORMATION_SCHEMA].[TABLES]
                          Where     [TABLE_SCHEMA] = 'Lookups'
                                    And [TABLE_NAME] = 'GenTransactionSource' )
           )
            Begin
                Create Table [Lookups].[GenTransactionSource]
                    (
                      [Source] Char(2)
                    , [SourceDesc] Varchar(250)
                    , [LastUpdated] DateTime2
                    );
            End;


        Declare @LastUpdate DateTime2 = GetDate();
        Declare @LastDate DateTime2;

        Select  @LastDate = Max([bt].[LastUpdated])
        From    [Lookups].[GenTransactionSource] As [bt];

        If DateDiff(Minute , @LastDate , GetDate()) > ( @HoursBetweenUpdates
                                                        * 60 )
            Begin
                Insert  [Lookups].[GenTransactionSource]
```

Author: Johnson, Chris

```
                        ( [Source]
                        , [SourceDesc]
                        , [LastUpdated]
                        )
                        Select  *
                        From    ( Select    [Source] = 'JE'
                                          , [SourceDesc] = 'Normal Journal'
                                          , [LastUpdated] = @LastUpdate
                                  Union
                                  Select    [Source] = 'IC'
                                          , [SourceDesc] = 'Inter-company journal'
                                          , [LastUpdated] = @LastUpdate
                                  Union
                                  Select    [Source] = 'RV'
                                          , [SourceDesc] = 'Reversing journal'
                                          , [LastUpdated] = @LastUpdate
                                  Union
                                  Select    [Source] = 'YE'
                                          , [SourceDesc] = 'Year end closing'
                                          , [LastUpdated] = @LastUpdate
                                  Union
                                  Select    [Source] = 'PE'
                                          , [SourceDesc] = 'Period end journal'
                                          , [LastUpdated] = @LastUpdate
                                  Union
                                  Select    [Source] = 'AU'
                                          , [SourceDesc] = 'Auditor''s Adjustment'
                                          , [LastUpdated] = @LastUpdate
                                  Union
                                  Select    [Source] = 'HM'
                                          , [SourceDesc] = 'History Maintenance'
                                          , [LastUpdated] = @LastUpdate
                                  Union
                                  Select    [Source] = 'AP'
                                          , [SourceDesc] = 'Accounts Payable'
                                          , [LastUpdated] = @LastUpdate
                                  Union
                                  Select    [Source] = 'AR'
                                          , [SourceDesc] = 'A/R payments'
                                          , [LastUpdated] = @LastUpdate
                                  Union
                                  Select    [Source] = 'IN'
                                          , [SourceDesc] = 'Inventory'
                                          , [LastUpdated] = @LastUpdate
                                  Union
                                  Select    [Source] = 'GR'
                                          , [SourceDesc] = 'GRN system'
                                          , [LastUpdated] = @LastUpdate
                                  Union
                                  Select    [Source] = 'SA'
                                          , [SourceDesc] = 'A/R Sales'
                                          , [LastUpdated] = @LastUpdate
                                  Union
                                  Select    [Source] = 'AS'
                                          , [SourceDesc] = 'Assets'
                                          , [LastUpdated] = @LastUpdate
                                  Union
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Process.UspUpdate_-GenTransactionSource

```
                                    Select    [Source] = 'PA'
                                            , [SourceDesc] = 'Payroll'
                                            , [LastUpdated] = @LastUpdate
                                    Union
                                    Select    [Source] = 'WP'
                                            , [SourceDesc] = 'Work in progress'
                                            , [LastUpdated] = @LastUpdate
                                    Union
                                    Select    [Source] = 'CS'
                                            , [SourceDesc] = 'Cash book'
                                            , [LastUpdated] = @LastUpdate
                                ) [t];


                If @PrevCheck = 1
                    Begin
                        Declare @CurrentCount Int
                          , @PreviousCount Int;

                        Select  @CurrentCount = Count(*)
                        From    [Lookups].[GenTransactionSource]
                        Where   [LastUpdated] = @LastUpdate;

                        Select  @PreviousCount = Count(*)
                        From    [Lookups].[GenTransactionSource]
                        Where   [LastUpdated] <> @LastUpdate;

                        If @PreviousCount > @CurrentCount
                            Begin
                                Delete  [Lookups].[GenTransactionSource]
                                Where   [LastUpdated] = @LastUpdate;
                                Print 'UspUpdate_GenTransactionSource - Count has
gone down since last run, no update applied';
                                Print 'Current Count = '
                                    + Cast(@CurrentCount As Varchar(5))
                                    + ' Previous Count = '
                                    + Cast(@PreviousCount As Varchar(5));
                            End;
                        If @PreviousCount <= @CurrentCount
                            Begin
                                Delete  [Lookups].[GenTransactionSource]
                                Where   [LastUpdated] <> @LastUpdate;
                                Print 'UspUpdate_GenTransactionSource - Update
applied successfully';
                            End;
                    End;
                If @PrevCheck = 0
                    Begin
                        Delete  [Lookups].[GenTransactionSource]
                        Where   [LastUpdated] <> @LastUpdate;
                        Print 'UspUpdate_GenTransactionSource - Update applied
successfully';
                    End;
            End;
        If DateDiff(Minute , @LastDate , GetDate()) <= ( @HoursBetweenUpdates
                                                          * 60 )
            Begin
                Print 'UspUpdate_GenTransactionSource - Table was last updated at '
```

Author: Johnson, Chris

```
                    + Cast(@LastDate As Varchar(255)) + ' no update applied';
            End;
    End;
GO
EXEC sp_addextendedproperty N'MS_Description', N'Stored proc to update specified
table', 'SCHEMA', N'Process', 'PROCEDURE', N'UspUpdate_GenTransactionSource', NULL,
NULL
GO
```

Uses

[Lookups].[GenTransactionSource]
[Process]

Author: Johnson, Chris

[Process].[UspUpdate_GlAccountCode]

MS_Description

Stored proc to update specified table

Parameters

Name	Data Type	Max Length (Bytes)
@PrevCheck	int	4
@HoursBetweenUpdates	numeric(5,2)	5

SQL Script

```
CREATE Proc [Process].[UspUpdate_GlAccountCode]
    (
      @PrevCheck Int --if count is less than previous don't update

    , @HoursBetweenUpdates Numeric(5 , 2)
    )
As
    Begin
/*
Stored procedure created by Chris Johnson, Prometic Group September 2015 to populate
table with amounts relating to    GL Account Code
*/

        Set NoCount On;


--check if table exists and create if it doesn't

        If ( Not Exists ( Select    1
                          From      [INFORMATION_SCHEMA].[TABLES]
                          Where     [TABLE_SCHEMA] = 'Lookups'
                                    And [TABLE_NAME] = 'GlAccountCode' )
           )
            Begin
                Create Table [Lookups].[GlAccountCode]
                    (
                      [Company] Varchar(150)                collate
latin1_general_bin
                    , [GlAccountCode] Char(5)                collate
latin1_general_bin
                    , [GlAccountDescription] Varchar(150)    collate
latin1_general_bin
                    , [LastUpdated] DateTime2
                    );
            End;


--check last time run and update if it's been longer than @HoursBetweenUpdates hours
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Process.UspUpdate_Gl-AccountCode

```
        Declare @LastDate DateTime2;

        Select  @LastDate = Max([LastUpdated])
        From    [Lookups].[GlAccountCode];

        If @LastDate Is Null
            Or DateDiff(Minute , @LastDate , GetDate()) > ( @HoursBetweenUpdates
                                                            * 60 )
            Begin
    --Set time of run

                Declare @LastUpdated DateTime2;
                Select  @LastUpdated = GetDate();

    --create master list of how codes affect stock

                Create Table [#OrdersGlAccountCode]
                    (
                      [GlAccountCode] Varchar(5)            collate
latin1_general_bin
                    , [GlAccountDescription] Varchar(150)    collate
latin1_general_bin
                    );

                Insert  [#OrdersGlAccountCode]
                        ( [GlAccountCode]
                        , [GlAccountDescription]
                        )
                        Select  [t].[GlAccountCode]
                              , [t].[GlAccountDescription]
                        From    ( Select    [GlAccountCode] = 'M'
                                          , [GlAccountDescription] = 'Merchandise
expense'
                                  Union
                                  Select    [GlAccountCode] = 'F'
                                          , [GlAccountDescription] = 'Freight-in
expense'
                                  Union
                                  Select    [GlAccountCode] = 'O'
                                          , [GlAccountDescription] = 'Other expense'
                                ) [t];

    --create script to pull data from each db into the tables

                Declare @SQL Varchar(Max) = 'USE [?];
        Declare @DB varchar(150),@DBCode varchar(150)
        Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end
        IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
        BEGIN
        Insert #Table1
            ( CompanyName )
        Select @DBCode
        End';

    --all companies process the same way

                Select  [CompanyName] = '40'
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Process.UspUpdate_Gl-AccountCode

```
                      , [O].[GlAccountCode]
                      , [O].[GlAccountDescription]
                Into    [#ResultsGlAccountCode]
                From    [#OrdersGlAccountCode] [O];

                Insert  [Lookups].[GlAccountCode]
                        ( [Company]
                        , [GlAccountCode]
                        , [GlAccountDescription]
                        , [LastUpdated]
                        )
                        Select  [CompanyName]
                              , [GlAccountCode]
                              , [GlAccountDescription]
                              , @LastUpdated
                        From    [#ResultsGlAccountCode];

                If @PrevCheck = 1
                    Begin
                        Declare @CurrentCount Int
                          , @PreviousCount Int;

                        Select  @CurrentCount = Count(*)
                        From    [Lookups].[GlAccountCode]
                        Where   [LastUpdated] = @LastUpdated;

                        Select  @PreviousCount = Count(*)
                        From    [Lookups].[GlAccountCode]
                        Where   [LastUpdated] <> @LastUpdated;

                        If @PreviousCount > @CurrentCount
                            Begin
                                Delete  [Lookups].[GlAccountCode]
                                Where   [LastUpdated] = @LastUpdated;
                                Print 'UspUpdate_GlAccountCode - Count has gone down
since last run, no update applied';
                                Print 'Current Count = '
                                    + Cast(@CurrentCount As Varchar(5))
                                    + ' Previous Count = '
                                    + Cast(@PreviousCount As Varchar(5));
                            End;
                        If @PreviousCount <= @CurrentCount
                            Begin
                                Delete  [Lookups].[GlAccountCode]
                                Where   [LastUpdated] <> @LastUpdated;
                                Print 'UspUpdate_GlAccountCode - Update applied
successfully';
                            End;
                    End;
                If @PrevCheck = 0
                    Begin
                        Delete  [Lookups].[GlAccountCode]
                        Where   [LastUpdated] <> @LastUpdated;
                        Print 'UspUpdate_GlAccountCode - Update applied
successfully';
                    End;
            End;
    End;
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Process.UspUpdate_Gl-AccountCode

```
    If DateDiff(Minute , @LastDate , GetDate()) <= ( @HoursBetweenUpdates * 60 )
        Begin
            Print 'UspUpdate_GlAccountCode - Table was last updated at '
                + Cast(@LastDate As Varchar(255)) + ' no update applied';
        End;
GO
EXEC sp_addextendedproperty N'MS_Description', N'Stored proc to update specified
table', 'SCHEMA', N'Process', 'PROCEDURE', N'UspUpdate_GlAccountCode', NULL, NULL
GO
```

Uses

[Lookups].[GlAccountCode]
[Process]

Author: Johnson, Chris

[Process].[UspUpdate_GLAccountType]

MS_Description

Stored proc to update specified table

Parameters

Name	Data Type	Max Length (Bytes)
@PrevCheck	int	4
@HoursBetweenUpdates	numeric(5,2)	5

SQL Script

```
CREATE Proc [Process].[UspUpdate_GLAccountType]
    (
      @PrevCheck Int --if count is less than previous don't update

    , @HoursBetweenUpdates Numeric(5 , 2)
    )
As
    Begin

--check if table exists and create if it doesn't

        If ( Not Exists ( Select    1
                          From      [INFORMATION_SCHEMA].[TABLES]
                          Where     [TABLE_SCHEMA] = 'Lookups'
                                    And [TABLE_NAME] = 'GLAccountType' )
           )
            Begin
                Create Table [Lookups].[GLAccountType]
                    (
                      [GLAccountType] Varchar(10)        collate Latin1_General_BIN
                    , [GLAccountTypeDesc] Varchar(250)    collate Latin1_General_BIN
                    , [LastUpdated] DateTime2
                    );
            End;


--check last time run and update if it's been longer than @HoursBetweenUpdates hours

        Declare @LastDate DateTime2;

        Select  @LastDate = Max([LastUpdated])
        From    [Lookups].[GLAccountType];

        If @LastDate Is Null
            Or DateDiff(Minute , @LastDate , GetDate()) > ( @HoursBetweenUpdates
                                                            * 60 )
            Begin
    --Set time of run
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Process.UspUpdate_-
GLAccountType

```
                Declare @LastUpdated DateTime2;
                Select  @LastUpdated = GetDate();
    --create master list of how codes affect stock

                Create Table [#GLAccountType]
                    (
                      [AccountType] Varchar(10)                    collate Latin1_-
General_BIN
                    , [AccountTypeDescription] Varchar(250)        collate Latin1_-
General_BIN
                    );

                Insert  [#GLAccountType]
                        ( [AccountType]
                        , [AccountTypeDescription]
                        )
                        Select  [t].[AccountType]
                              , [t].[AccountTypeDescription]
                        From    ( Select    [AccountType] = 'A'
                                          , [AccountTypeDescription] = 'Asset'
                                  Union
                                  Select    [AccountType] = 'C'
                                          , [AccountTypeDescription] = 'Capital'
                                  Union
                                  Select    [AccountType] = 'E'
                                          , [AccountTypeDescription] = 'Expense'
                                  Union
                                  Select    [AccountType] = 'L'
                                          , [AccountTypeDescription] = 'Liability'
                                  Union
                                  Select    [AccountType] = 'R'
                                          , [AccountTypeDescription] = 'Revenue'
                                  Union
                                  Select    [AccountType] = 'S'
                                          , [AccountTypeDescription] = 'Statistical'
                                  Union
                                  Select    [AccountType] = 'T'
                                          , [AccountTypeDescription] = 'Template'
                                ) [t];



    --all companies process the same way

                Select  [AccountType] = Coalesce([GAT].[AccountType] ,
                                                 [GM].[AccountType])
                      , [AccountTypeDescription] = Coalesce([GAT].[AccountType-
Description] ,
                                                            'Unknown')
                Into    [#ResultsAccountTypeName]
                From    [#GLAccountType] As [GAT]
                        Full Outer Join [SysproCompany40].[dbo].[GenMaster] As [GM]
                            On [GM].[AccountType] = [GAT].[AccountType]
                Group By Coalesce([GAT].[AccountType] , [GM].[AccountType])
                      , Coalesce([AccountTypeDescription] , 'Unknown');
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Process.UspUpdate_-GLAccountType

```
    --placeholder for anomalous results that are different to master list


                Insert  [Lookups].[GLAccountType]
                        ( [GLAccountType]
                        , [GLAccountTypeDesc]
                        , [LastUpdated]
                        )
                        Select  [rcn].[AccountType]
                              , [rcn].[AccountTypeDescription]
                              , @LastUpdated
                        From    [#ResultsAccountTypeName] As [rcn];

                If @PrevCheck = 1
                    Begin
                        Declare @CurrentCount Int
                          , @PreviousCount Int;

                        Select  @CurrentCount = Count(*)
                        From    [Lookups].[GLAccountType] As [cn]
                        Where   [cn].[LastUpdated] = @LastUpdated;

                        Select  @PreviousCount = Count(*)
                        From    [Lookups].[GLAccountType] As [cn]
                        Where   [cn].[LastUpdated] <> @LastDate;

                        If @PreviousCount > @CurrentCount
                            Begin
                                Delete  [Lookups].[GLAccountType]
                                Where   [LastUpdated] = @LastDate;
                                Print 'UspUpdate_GLAccountType - Count has gone down
since last run, no update applied';
                                Print 'Current Count = '
                                    + Cast(@CurrentCount As Varchar(5))
                                    + ' Previous Count = '
                                    + Cast(@PreviousCount As Varchar(5));
                            End;
                        If @PreviousCount <= @CurrentCount
                            Begin
                                Delete  [Lookups].[GLAccountType]
                                Where   [LastUpdated] <> @LastUpdated
                                        Or [LastUpdated] Is Null;
                                Print 'UspUpdate_GLAccountType - Update applied
successfully';
                            End;
                    End;
                If @PrevCheck = 0
                    Begin
                        Delete  [Lookups].[GLAccountType]
                        Where   [LastUpdated] <> @LastUpdated;
                        Print 'UspUpdate_GLAccountType - Update applied
successfully';
                    End;
            End;
    End;
    If DateDiff(Minute , @LastDate , GetDate()) <= ( @HoursBetweenUpdates * 60 )
        Begin
            Print 'UspUpdate_GLAccountType - Table was last updated at '
```

Author: Johnson, Chris

```
                + Cast(@LastDate As Varchar(255)) + ' no update applied';
        End;
GO
EXEC sp_addextendedproperty N'MS_Description', N'Stored proc to update specified
table', 'SCHEMA', N'Process', 'PROCEDURE', N'UspUpdate_GLAccountType', NULL, NULL
GO
```

Uses

[Lookups].[GLAccountType]
[Process]

Author: Johnson, Chris

[Process].[UspUpdate_GlAnalysisCategories]

MS_Description

Stored proc to update specified table

Parameters

Name	Data Type	Max Length (Bytes)
@PrevCheck	int	4
@HoursBetweenUpdates	numeric(5,2)	5

SQL Script

```
CREATE Proc [Process].[UspUpdate_GlAnalysisCategories]
    (
      @PrevCheck Int
    , @HoursBetweenUpdates Numeric(5 , 2)
    )
As
    Begin

--remove nocount on to speed up query

        Set NoCount On;

--check if table exists and create if it doesn't

        If ( Not Exists ( Select    1
                          From      [INFORMATION_SCHEMA].[TABLES]
                          Where     [TABLE_SCHEMA] = 'Lookups'
                                    And [TABLE_NAME] = 'GlAnalysisCategory' )
           )
            Begin
                Create Table [Lookups].[GlAnalysisCategory]
                    (
                      [Company] Varchar(150)            collate latin1_general_bin
                    , [GlAnalysisCategory] Varchar(150)    collate
latin1_general_bin
                    , [LastUpdated] DateTime2
                    );
            End;



--check last time run and update if it's been longer than @HoursBetweenUpdates hours

        Declare @LastDate DateTime2;

        Select  @LastDate = Max([LastUpdated])
        From    [Lookups].[GlAnalysisCategory];
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Process.UspUpdate_Gl-AnalysisCategories

```
        If @LastDate Is Null
            Or DateDiff(Minute , @LastDate , GetDate()) > ( @HoursBetweenUpdates
                                                            * 60 )
            Begin
    --Set time of run

                Declare @LastUpdated DateTime2;
                Select  @LastUpdated = GetDate();

--list the tables that are to be pulled back from each DB - if they are not found
the script will not be run against that db

                Declare @ListOfTables Varchar(Max) = 'GenAnalysisTrn';

--create temporary tables to be pulled from different databases, including a column
to id

                Create Table [#Table1GLA]
                    (
                      [Company] Varchar(150)            collate latin1_general_bin
                    , [GlAnalysisCategory] Varchar(150)    collate
latin1_general_bin
                    );

--create script to pull data from each db into the tables

                Declare @Company Varchar(30) = 'All';
                Declare @SQL Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
                    + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables
                    + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'',''))
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
                Insert #Table1GLA
                    ( Company, [GlAnalysisCategory])
                Select distinct @DBCode
                ,[AnalysisCategory]
                From [GenAnalysisTrn]
            End
    End';


--execute script against each db, populating the base tables
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Process.UspUpdate_Gl-AnalysisCategories

```
                Exec [Process].[ExecForEachDB] @cmd = @SQL;

                Insert  [Lookups].[GlAnalysisCategory]
                        ( [Company]
                        , [GlAnalysisCategory]
                        , [LastUpdated]
                        )
                        Select  [Company]
                              , [GlAnalysisCategory]
                              , @LastUpdated
                        From    [#Table1GLA];

                If @PrevCheck = 1
                    Begin
                        Declare @CurrentCount Int
                          , @PreviousCount Int;

                        Select  @CurrentCount = Count(*)
                        From    [Lookups].[GlAnalysisCategory]
                        Where   [LastUpdated] = @LastUpdated;

                        Select  @PreviousCount = Count(*)
                        From    [Lookups].[GlAnalysisCategory]
                        Where   [LastUpdated] <> @LastUpdated;

                        If @PreviousCount > @CurrentCount
                            Begin
                                Delete  [Lookups].[GlAnalysisCategory]
                                Where   [LastUpdated] = @LastUpdated;
                                Print 'UspUpdate_GlAnalysisCategories - Count has
gone down since last run, no update applied';
                                Print 'Current Count = '
                                    + Cast(@CurrentCount As Varchar(5))
                                    + ' Previous Count = '
                                    + Cast(@PreviousCount As Varchar(5));
                            End;
                        If @PreviousCount <= @CurrentCount
                            Begin
                                Delete  [Lookups].[GlAnalysisCategory]
                                Where   [LastUpdated] <> @LastUpdated;
                                Print 'UspUpdate_GlAnalysisCategories - Update
applied successfully';
                            End;
                    End;
                If @PrevCheck = 0
                    Begin
                        Delete  [Lookups].[GlAnalysisCategory]
                        Where   [LastUpdated] <> @LastUpdated;
                        Print 'UspUpdate_GlAnalysisCategories - Update applied
successfully';
                    End;
            End;
    End;
    If DateDiff(Minute , @LastDate , GetDate()) <= ( @HoursBetweenUpdates * 60 )
        Begin
            Print 'UspUpdate_GlAnalysisCategories - Table was last updated at '
                + Cast(@LastDate As Varchar(255)) + ' no update applied';
        End;
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Process.UspUpdate_Gl-AnalysisCategories

```
GO
EXEC sp_addextendedproperty N'MS_Description', N'Stored proc to update specified
table', 'SCHEMA', N'Process', 'PROCEDURE', N'UspUpdate_GlAnalysisCategories', NULL,
NULL
GO
```

Uses

[Lookups].[GlAnalysisCategory]
[Process].[ExecForEachDB]
[Process]

Author: Johnson, Chris

[Process].[UspUpdate_GlExpenseCode]

MS_Description

Stored proc to update specified table

Parameters

Name	Data Type	Max Length (Bytes)
@PrevCheck	int	4
@HoursBetweenUpdates	numeric(5,2)	5

SQL Script

```
CREATE Proc [Process].[UspUpdate_GlExpenseCode]
    (
      @PrevCheck Int --if count is less than previous don't update

    , @HoursBetweenUpdates Numeric(5 , 2)
    )
As
    Begin

--check if table exists and create if it doesn't

        If ( Not Exists ( Select    1
                          From      [INFORMATION_SCHEMA].[TABLES]
                          Where     [TABLE_SCHEMA] = 'Lookups'
                                    And [TABLE_NAME] = 'GlExpenseCode' )
           )
            Begin
                Create Table [Lookups].[GlExpenseCode]
                    (
                      [Company] Varchar(150)                collate Latin1_General_-
BIN
                    , [GlExpenseCode] Char(5)                collate Latin1_General_-
BIN
                    , [GlExpenseDescription] Varchar(150)    collate Latin1_General_-
BIN
                    , [LastUpdated] DateTime2
                    );
            End;


--check last time run and update if it's been longer than @HoursBetweenUpdates hours

        Declare @LastDate DateTime2;

        Select  @LastDate = Max([LastUpdated])
        From    [Lookups].[GlExpenseCode];

        If @LastDate Is Null
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Process.UspUpdate_Gl-ExpenseCode

```
            Or DateDiff(Minute , @LastDate , GetDate()) > ( @HoursBetweenUpdates
                                                            * 60 )
            Begin
    --Set time of run

                Declare @LastUpdated DateTime2;
                Select  @LastUpdated = GetDate();

    --create master list of how codes affect stock

                Create Table [#OrdersGlExpenseCode]
                    (
                      [GlExpenseCode] Varchar(5)            collate Latin1_General_-
BIN
                    , [GlExpenseDescription] Varchar(150)    collate Latin1_General_-
BIN
                    );

                Insert  [#OrdersGlExpenseCode]
                        ( [GlExpenseCode]
                        , [GlExpenseDescription]
                        )
                        Select  [t].[GlExpenseCode]
                              , [t].[GlExpenseDescription]
                        From    ( Select    [GlExpenseCode] = 'M'
                                          , [GlExpenseDescription] = 'Merchandise
expense'
                                  Union
                                  Select    [GlExpenseCode] = 'F'
                                          , [GlExpenseDescription] = 'Freight-in
expense'
                                  Union
                                  Select    [GlExpenseCode] = 'O'
                                          , [GlExpenseDescription] = 'Other expense'
                                ) [t];


    --create script to pull data from each db into the tables

                Declare @SQL Varchar(Max) = 'USE [?];
        Declare @DB varchar(150),@DBCode varchar(150)
        Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end
        IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
        BEGIN
        Insert #Table1
            ( CompanyName )
        Select @DBCode
        End';

    --all companies process the same way

                Select  [CompanyName] = '40'
                      , [O].[GlExpenseCode]
                      , [O].[GlExpenseDescription]
                Into    [#ResultsGlExpenseCode]
                From    [#OrdersGlExpenseCode] [O];
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Process.UspUpdate_Gl-ExpenseCode

```
    --placeholder for anomalous results that are different to master list


                Insert  [Lookups].[GlExpenseCode]
                        ( [Company]
                        , [GlExpenseCode]
                        , [GlExpenseDescription]
                        , [LastUpdated]
                        )
                        Select  [CompanyName]
                              , [GlExpenseCode]
                              , [GlExpenseDescription]
                              , @LastUpdated
                        From    [#ResultsGlExpenseCode];

                If @PrevCheck = 1
                    Begin
                        Declare @CurrentCount Int
                          , @PreviousCount Int;

                        Select  @CurrentCount = Count(*)
                        From    [Lookups].[GlExpenseCode]
                        Where   [LastUpdated] = @LastUpdated;

                        Select  @PreviousCount = Count(*)
                        From    [Lookups].[GlExpenseCode]
                        Where   [LastUpdated] <> @LastUpdated;

                        If @PreviousCount > @CurrentCount
                            Begin
                                Delete  [Lookups].[GlExpenseCode]
                                Where   [LastUpdated] = @LastUpdated;
                                Print 'UspUpdate_GlExpenseCode - Count has gone down
since last run, no update applied';
                                Print 'Current Count = '
                                    + Cast(@CurrentCount As Varchar(5))
                                    + ' Previous Count = '
                                    + Cast(@PreviousCount As Varchar(5));
                            End;
                        If @PreviousCount <= @CurrentCount
                            Begin
                                Delete  [Lookups].[GlExpenseCode]
                                Where   [LastUpdated] <> @LastUpdated;
                                Print 'UspUpdate_GlExpenseCode - Update applied
successfully';
                            End;
                    End;
                If @PrevCheck = 0
                    Begin
                        Delete  [Lookups].[GlExpenseCode]
                        Where   [LastUpdated] <> @LastUpdated;
                        Print 'UspUpdate_GlExpenseCode - Update applied
successfully';
                    End;
            End;
    End;
    If DateDiff(Minute , @LastDate , GetDate()) <= ( @HoursBetweenUpdates * 60 )
```

Author: Johnson, Chris

```
        Begin
            Print 'UspUpdate_GlExpenseCode - Table was last updated at '
                + Cast(@LastDate As Varchar(255)) + ' no update applied';
        End;
GO
EXEC sp_addextendedproperty N'MS_Description', N'Stored proc to update specified
table', 'SCHEMA', N'Process', 'PROCEDURE', N'UspUpdate_GlExpenseCode', NULL, NULL
GO
```

Uses

[Lookups].[GlExpenseCode]
[Process]

[Process].[UspUpdate_InvMaster_PartCategory]

MS_Description

Stored proc to update specified table

Parameters

Name	Data Type	Max Length (Bytes)
@PrevCheck	int	4
@HoursBetweenUpdates	numeric(5,2)	5

SQL Script

```
CREATE Proc [Process].[UspUpdate_InvMaster_PartCategory]
    (
      @PrevCheck Int --if count is less than previous don't update

    , @HoursBetweenUpdates Numeric(5 , 2)
    )
As
    Begin

--check if table exists and create if it doesn't

        If ( Not Exists ( Select    1
                          From      [INFORMATION_SCHEMA].[TABLES]
                          Where     [TABLE_SCHEMA] = 'Lookups'
                                    And [TABLE_NAME] = 'InvMaster_PartCategory' )
           )
            Begin
                Create Table [Lookups].[InvMaster_PartCategory]
                    (
                      [PartCategoryCode] Varchar(10)
                        Collate Latin1_General_BIN
                    , [PartCategoryDescription] Varchar(150)
                        Collate Latin1_General_BIN
                    , [LastUpdated] DateTime2
                    );
            End;


--check last time run and update if it's been longer than @HoursBetweenUpdates hours

        Declare @LastDate DateTime2;

        Select  @LastDate = Max([LastUpdated])
        From    [Lookups].[PurchaseOrderStatus];

        If @LastDate Is Null
            Or DateDiff(Minute , @LastDate , GetDate()) > ( @HoursBetweenUpdates
                                                            * 60 )
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Process.UspUpdate_Inv-Master_PartCategory

```
            Begin
    --Set time of run

                Declare @LastUpdated DateTime2;
                Select  @LastUpdated = GetDate();
    --create master list of how codes affect stock

                Create Table [#PartCategoryList]
                    (
                      [PartCategory] Varchar(150) Collate Latin1_General_BIN
                    , [PartCategoryName] Varchar(250)
                      Collate Latin1_General_BIN
                    );

                Insert  [#PartCategoryList]
                        ( [PartCategory]
                        , [PartCategoryName]
                        )
                        Select  [t].[PartCategory]
                              , [t].[PartCategoryName]
                        From    ( Select    [PartCategory] = 'M'
                                          , [PartCategoryName] = 'Made-in'
                                  Union
                                  Select    [PartCategory] = 'B'
                                          , [PartCategoryName] = 'Bought Out'
                                  Union
                                  Select    [PartCategory] = 'S'
                                          , [PartCategoryName] = 'Sub-contracted'
                                  Union
                                  Select    [PartCategory] = 'G'
                                          , [PartCategoryName] = 'Phantom/Ghost
Part'
                                  Union
                                  Select    [PartCategory] = 'P'
                                          , [PartCategoryName] = 'Planning bill'
                                  Union
                                  Select    [PartCategory] = 'K'
                                          , [PartCategoryName] = 'Kit Part'
                                  Union
                                  Select    [PartCategory] = 'C'
                                          , [PartCategoryName] = 'Co-Product'
                                  Union
                                  Select    [PartCategory] = 'Y'
                                          , [PartCategoryName] = 'By-product'
                                  Union
                                  Select    [PartCategory] = 'N'
                                          , [PartCategoryName] = 'Notional part'
                                ) [t];


                Insert  [Lookups].[InvMaster_PartCategory]
                        ( [PartCategoryCode]
                        , [PartCategoryDescription]
                        , [LastUpdated]
                        )
                        Select  [CL].[PartCategory]
                              , [CL].[PartCategoryName]
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Process.UspUpdate_Inv-Master_PartCategory

```
                                , @LastUpdated
                        From    [#PartCategoryList] As [CL];

                If @PrevCheck = 1
                    Begin
                        Declare @CurrentCount Int
                          , @PreviousCount Int;

                        Select  @CurrentCount = Count(*)
                        From    [Lookups].[InvMaster_PartCategory] As [cn]
                        Where   [cn].[LastUpdated] = @LastUpdated;

                        Select  @PreviousCount = Count(*)
                        From    [Lookups].[InvMaster_PartCategory] As [cn]
                        Where   [cn].[LastUpdated] <> @LastDate;

                        If @PreviousCount > @CurrentCount
                            Begin
                                Delete  [Lookups].[InvMaster_PartCategory]
                                Where   [LastUpdated] = @LastDate;
                                Print 'UspUpdate_InvMaster_PartCategory - Count has
gone down since last run, no update applied';
                                Print 'Current Count = '
                                    + Cast(@CurrentCount As Varchar(5))
                                    + ' Previous Count = '
                                    + Cast(@PreviousCount As Varchar(5));
                            End;
                        If @PreviousCount <= @CurrentCount
                            Begin
                                Delete  [Lookups].[InvMaster_PartCategory]
                                Where   [LastUpdated] <> @LastUpdated
                                        Or [LastUpdated] Is Null;
                                Print 'UspUpdate_InvMaster_PartCategory - Update
applied successfully';
                            End;
                    End;
                If @PrevCheck = 0
                    Begin
                        Delete  [Lookups].[InvMaster_PartCategory]
                        Where   [LastUpdated] <> @LastUpdated;
                        Print 'UspUpdate_InvMaster_PartCategory - Update applied
successfully';
                    End;
            End;
    End;
    If DateDiff(Minute , @LastDate , GetDate()) <= ( @HoursBetweenUpdates * 60 )
        Begin
            Print 'UspUpdate_InvMaster_PartCategory - Table was last updated at '
                + Cast(@LastDate As Varchar(255)) + ' no update applied';
        End;
GO
EXEC sp_addextendedproperty N'MS_Description', N'Stored proc to update specified
table', 'SCHEMA', N'Process', 'PROCEDURE', N'UspUpdate_InvMaster_PartCategory',
NULL, NULL
GO
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Process.UspUpdate_Inv-Master_PartCategory

Uses

[Lookups].[InvMaster_PartCategory]

[Lookups].[PurchaseOrderStatus]

[Process]

Author: Johnson, Chris

[Process].[UspUpdate_MCompleteFlag]

MS_Description

Stored proc to update specified table

Parameters

Name	Data Type	Max Length (Bytes)
@PrevCheck	int	4
@HoursBetweenUpdates	numeric(5,2)	5

SQL Script

```
CREATE Proc [Process].[UspUpdate_MCompleteFlag]
    (
      @PrevCheck Int --if count is less than previous don't update

    , @HoursBetweenUpdates Numeric(5 , 2)
    )
As
    Begin
        Set NoCount On;

--check if table exists and create if it doesn't

        If ( Not Exists ( Select    1
                          From      [INFORMATION_SCHEMA].[TABLES]
                          Where     [TABLE_SCHEMA] = 'Lookups'
                                    And [TABLE_NAME] = 'MCompleteFlag' )
           )
            Begin
                Create Table [Lookups].[MCompleteFlag]
                    (
                      [Company] Varchar(150) Collate Latin1_General_BIN
                    , [MCompleteFlagCode] Char(5) Collate Latin1_General_BIN
                    , [MCompleteFlagDescription] Varchar(150) Collate Latin1_-
General_BIN
                    , [LastUpdated] DateTime2
                    );
            End;


--check last time run and update if it's been longer than @HoursBetweenUpdates hours

        Declare @LastDate DateTime2;

        Select  @LastDate = Max([LastUpdated])
        From    [Lookups].[MCompleteFlag];

        If @LastDate Is Null
            Or DateDiff(Minute , @LastDate , GetDate()) > ( @HoursBetweenUpdates
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Process.UspUpdate_-MCompleteFlag

```
                                                        * 60 )
            Begin
    --Set time of run

                Declare @LastUpdated DateTime2;
                Select  @LastUpdated = GetDate();

    --create master list of how codes affect stock

                Create Table [#Orders]
                    (
                      [MCompleteFlagCode] Varchar(5) Collate Latin1_General_BIN
                    , [MCompleteFlagDescription] Varchar(150) Collate Latin1_-
General_BIN
                    );

                Insert  [#Orders]
                        ( [MCompleteFlagCode]
                        , [MCompleteFlagDescription]
                        )
                        Select  [t].[MCompleteFlagCode]
                              , [t].[MCompleteFlagDescription]
                        From    ( Select    [MCompleteFlagCode] = 'Y'
                                          , [MCompleteFlagDescription] = 'Yes'
                                  Union
                                  Select    [MCompleteFlagCode] = 'N'
                                          , [MCompleteFlagDescription] = 'No'
                                  Union
                                  Select    [MCompleteFlagCode] = ' '
                                          , [MCompleteFlagDescription] = 'No'
                                ) [t];

    --Get list of all companies in use


    --create temporary tables to be pulled from different databases, including a
column to id

                Create Table [#Table1]
                    (
                      [CompanyName] Varchar(150)
                    );

    --create script to pull data from each db into the tables

                Declare @SQL Varchar(Max) = 'USE [?];
        Declare @DB varchar(150),@DBCode varchar(150)
        Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end
        IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
        BEGIN
        Insert #Table1
            ( CompanyName )
        Select @DBCode
        End';

    --execute script against each db, populating the base tables
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Process.UspUpdate_-MCompleteFlag

```
                Exec [Process].[ExecForEachDB] @cmd = @SQL;

    --all companies process the same way

                Select  [T].[CompanyName]
                      , [O].[MCompleteFlagCode]
                      , [O].[MCompleteFlagDescription]
                Into    [#ResultsMFlag]
                From    [#Table1] [T]
                        Left Join [#Orders] [O]
                            On 1 = 1;

    --placeholder for anomalous results that are different to master list


                Insert  [Lookups].[MCompleteFlag]
                        ( [Company]
                        , [MCompleteFlagCode]
                        , [MCompleteFlagDescription]
                        , [LastUpdated]
                        )
                        Select  [CompanyName]
                              , [MCompleteFlagCode]
                              , [MCompleteFlagDescription]
                              , @LastUpdated
                        From    [#ResultsMFlag];

                If @PrevCheck = 1
                    Begin
                        Declare @CurrentCount Int
                          , @PreviousCount Int;

                        Select  @CurrentCount = Count(*)
                        From    [Lookups].[MCompleteFlag]
                        Where   [LastUpdated] = @LastUpdated;

                        Select  @PreviousCount = Count(*)
                        From    [Lookups].[MCompleteFlag]
                        Where   [LastUpdated] <> @LastUpdated;

                        If @PreviousCount > @CurrentCount
                            Begin
                                Delete  [Lookups].[MCompleteFlag]
                                Where   [LastUpdated] = @LastUpdated;
                                Print 'UspUpdate_MCompleteFlag - Count has gone down
since last run, no update applied';
                                Print 'Current Count = '
                                    + Cast(@CurrentCount As Varchar(5))
                                    + ' Previous Count = '
                                    + Cast(@PreviousCount As Varchar(5));
                            End;
                        If @PreviousCount <= @CurrentCount
                            Begin
                                Delete  [Lookups].[MCompleteFlag]
                                Where   [LastUpdated] <> @LastUpdated;
                                Print 'UspUpdate_MCompleteFlag - Update applied
successfully';
```

Author: Johnson, Chris

```
                            End;
                    End;
                If @PrevCheck = 0
                    Begin
                        Delete  [Lookups].[MCompleteFlag]
                        Where   [LastUpdated] <> @LastUpdated;
                        Print 'UspUpdate_MCompleteFlag - Update applied
successfully';
                    End;
            End;
    End;
    If DateDiff(Minute , @LastDate , GetDate()) <= ( @HoursBetweenUpdates * 60 )
        Begin
            Print 'UspUpdate_MCompleteFlag - Table was last updated at '
                + Cast(@LastDate As Varchar(255)) + ' no update applied';
        End;
GO
EXEC sp_addextendedproperty N'MS_Description', N'Stored proc to update specified
table', 'SCHEMA', N'Process', 'PROCEDURE', N'UspUpdate_MCompleteFlag', NULL, NULL
GO
```

Uses

[Lookups].[MCompleteFlag]
[Process].[ExecForEachDB]
[Process]

Author: Johnson, Chris

[Process].[UspUpdate_PorLineType]

MS_Description

Stored proc to update specified table

Parameters

Name	Data Type	Max Length (Bytes)
@PrevCheck	int	4
@HoursBetweenUpdates	numeric(5,2)	5

SQL Script

```
CREATE Proc [Process].[UspUpdate_PorLineType]
    (
      @PrevCheck Int
    , @HoursBetweenUpdates Numeric(5 , 2)
    )
As
    Begin
--remove nocount on to speed up query

        Set NoCount On;

--check if table exists and create if it doesn't

        If ( Not Exists ( Select    1
                          From      [INFORMATION_SCHEMA].[TABLES]
                          Where     [TABLE_SCHEMA] = 'Lookups'
                                    And [TABLE_NAME] = 'PorLineType' )
           )
            Begin
                Create Table [Lookups].[PorLineType]
                    (
                      [PorLineType] Int Primary Key
                    , [PorLineTypeDesc] Varchar(150)    collate latin1_general_bin
                    , [LastUpdated] DateTime2
                    );
            End;



--check last time run and update if it's been longer than @HoursBetweenUpdates hours

        Declare @LastDate DateTime2;

        Select  @LastDate = Max([LastUpdated])
        From    [Lookups].[PorLineType];

        If @LastDate Is Null
            Or DateDiff(Minute , @LastDate , GetDate()) > ( @HoursBetweenUpdates
```

Author: Johnson, Chris

```
                                                        * 60 )
            Begin
    --Set time of run

                Declare @LastUpdated DateTime2;
                Select  @LastUpdated = GetDate();

--list the tables that are to be pulled back from each DB - if they are not found
the script will not be run against that db

                Declare @ListOfTables Varchar(Max) = 'InvMovements';

                Insert  [Lookups].[PorLineType]
                        ( [PorLineType]
                        , [LastUpdated]
                        , [PorLineTypeDesc]
                        )
                        Select  [t].[PorLineType]
                              , [LastUpdated] = @LastUpdated
                              , [t].[PorLineTypeDesc]
                        From    ( Select    [PorLineType] = 1
                                          , [PorLineTypeDesc] = 'Stocked line'
                                  Union
                                  Select    [PorLineType] = 4
                                          , [PorLineTypeDesc] = 'Freight line'
                                  Union
                                  Select    [PorLineType] = 5
                                          , [PorLineTypeDesc] = 'Other charges'
                                  Union
                                  Select    [PorLineType] = 6
                                          , [PorLineTypeDesc] = 'CommentLine'
                                  Union
                                  Select    [PorLineType] = 7
                                          , [PorLineTypeDesc] = 'Non-Stocked line'
                                ) [t];
                If @PrevCheck = 1
                    Begin
                        Declare @CurrentCount Int
                          , @PreviousCount Int;

                        Select  @CurrentCount = Count(*)
                        From    [Lookups].[PorLineType]
                        Where   [LastUpdated] = @LastUpdated;

                        Select  @PreviousCount = Count(*)
                        From    [Lookups].[PorLineType]
                        Where   [LastUpdated] <> @LastUpdated;

                        If @PreviousCount > @CurrentCount
                            Begin
                                Delete  [Lookups].[PorLineType]
                                Where   [LastUpdated] = @LastUpdated;
                                Print 'UspUpdate_PorLineType - Count has gone down
since last run, no update applied';
                                Print 'Current Count = '
                                    + Cast(@CurrentCount As Varchar(5))
                                    + ' Previous Count = '
```

Author: Johnson, Chris

```
                                    + Cast(@PreviousCount As Varchar(5));
                            End;
                        If @PreviousCount <= @CurrentCount
                            Begin
                                Delete  [Lookups].[PorLineType]
                                Where   [LastUpdated] <> @LastUpdated;
                                Print 'UspUpdate_PorLineType - Update applied
successfully';
                            End;
                    End;
                If @PrevCheck = 0
                    Begin
                        Delete  [Lookups].[PorLineType]
                        Where   [LastUpdated] <> @LastUpdated;
                        Print 'UspUpdate_PorLineType - Update applied successfully';
                    End;
            End;
    End;
    If DateDiff(Minute , @LastDate , GetDate()) <= ( @HoursBetweenUpdates * 60 )
        Begin
            Print 'UspUpdate_PorLineType - Table was last updated at '
                + Cast(@LastDate As Varchar(255)) + ' no update applied';
        End;

GO
EXEC sp_addextendedproperty N'MS_Description', N'Stored proc to update specified
table', 'SCHEMA', N'Process', 'PROCEDURE', N'UspUpdate_PorLineType', NULL, NULL
GO
```

Uses

[Lookups].[PorLineType]
[Process]

Author: Johnson, Chris

[Process].[UspUpdate_ProductClass]

MS_Description

Stored proc to update specified table

Parameters

Name	Data Type	Max Length (Bytes)
@PrevCheck	int	4
@HoursBetweenUpdates	numeric(5,2)	5

SQL Script

```
CREATE Proc [Process].[UspUpdate_ProductClass]
    (
      @PrevCheck Int
    , @HoursBetweenUpdates Numeric(5 , 2)
    )
As
    Begin

--remove nocount on to speed up query

        Set NoCount On;

--check if table exists and create if it doesn't

        If ( Not Exists ( Select    1
                          From      [INFORMATION_SCHEMA].[TABLES]
                          Where     [TABLE_SCHEMA] = 'Lookups'
                                    And [TABLE_NAME] = 'ProductClass' )
           )
            Begin
                Create Table [Lookups].[ProductClass]
                    (
                      [Company] Varchar(150) Collate Latin1_General_BIN
                    , [ProductClass] Varchar(150) Collate Latin1_General_BIN
                    , [ProductClassDescription] Varchar(250) Collate Latin1_General_-
BIN
                    , [LastUpdated] DateTime2
                    );
            End;



--check last time run and update if it's been longer than @HoursBetweenUpdates hours

        Declare @LastDate DateTime2;

        Select  @LastDate = Max([LastUpdated])
        From    [Lookups].[ProductClass];
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Process.UspUpdate_-ProductClass

```
        If @LastDate Is Null
            Or DateDiff(Minute , @LastDate , GetDate()) > ( @HoursBetweenUpdates
                                                            * 60 )
            Begin

--Set time of run

                Declare @LastUpdated DateTime2;
                Select  @LastUpdated = GetDate();

--list the tables that are to be pulled back from each DB - if they are not found
the script will not be run against that db

                Declare @ListOfTables Varchar(Max) = 'SalProductClass';

--create temporary tables to be pulled from different databases, including a column
to id

                Create Table [#Table1PC]
                    (
                      [Company] Varchar(150) Collate Latin1_General_BIN
                    , [ProductClass] Varchar(150) Collate Latin1_General_BIN
                    , [ProductClassDescription] Varchar(250) Collate Latin1_General_-
BIN
                    );

--create script to pull data from each db into the tables

                Declare @Company Varchar(30) = 'All';
                Declare @SQL Varchar(Max) = 'USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
                    + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables
                    + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'',''))
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
                Insert #Table1PC
                    ( Company, ProductClass, ProductClassDescription)
                Select
                    distinct @DBCode
                    , [ProductClass]
                    ,[Description]
                from dbo.SalProductClass

            End
```

Author: Johnson, Chris

```
    End';

--Enable this function to check script changes (try to run script directly against
db manually)

--Print @SQL


--execute script against each db, populating the base tables

                Exec [Process].[ExecForEachDB] @cmd = @SQL;

                Insert  [Lookups].[ProductClass]
                        ( [Company]
                        , [ProductClass]
                        , [ProductClassDescription]
                        , [LastUpdated]
                        )
                        Select  [Company]
                              , [ProductClass]
                              , [ProductClassDescription]
                              , @LastUpdated
                        From    [#Table1PC];

                If @PrevCheck = 1
                    Begin
                        Declare @CurrentCount Int
                          , @PreviousCount Int;

                        Select  @CurrentCount = Count(*)
                        From    [Lookups].[ProductClass]
                        Where   [LastUpdated] = @LastUpdated;

                        Select  @PreviousCount = Count(*)
                        From    [Lookups].[ProductClass]
                        Where   [LastUpdated] <> @LastUpdated;

                        If @PreviousCount > @CurrentCount
                            Begin
                                Delete  [Lookups].[ProductClass]
                                Where   [LastUpdated] = @LastUpdated;
                                Print 'UspUpdate_ProductClass - Count has gone down
since last run, no update applied';
                                Print 'Current Count = '
                                    + Cast(@CurrentCount As Varchar(5))
                                    + ' Previous Count = '
                                    + Cast(@PreviousCount As Varchar(5));
                            End;
                        If @PreviousCount <= @CurrentCount
                            Begin
                                Delete  [Lookups].[ProductClass]
                                Where   [LastUpdated] <> @LastUpdated;
                                Print 'UspUpdate_ProductClass - Update applied
successfully';
                            End;
                    End;
                If @PrevCheck = 0
                    Begin
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Process.UspUpdate_-ProductClass

```
                        Delete  [Lookups].[ProductClass]
                        Where   [LastUpdated] <> @LastUpdated;
                        Print 'UspUpdate_ProductClass - Update applied
successfully';
                    End;
            End;
    End;
    If DateDiff(Minute , @LastDate , GetDate()) <= ( @HoursBetweenUpdates * 60 )
        Begin
            Print 'UspUpdate_ProductClass - Table was last updated at '
                + Cast(@LastDate As Varchar(255)) + ' no update applied';
        End;


GO
EXEC sp_addextendedproperty N'MS_Description', N'Stored proc to update specified
table', 'SCHEMA', N'Process', 'PROCEDURE', N'UspUpdate_ProductClass', NULL, NULL
GO
```

Uses

[Lookups].[ProductClass]
[Process].[ExecForEachDB]
[Process]

Author: Johnson, Chris

[Process].[UspUpdate_PurchaseOrderInvoiceMapping]

MS_Description

Stored proc to update specified table

Parameters

Name	Data Type	Max Length (Bytes)
@PrevCheck	int	4
@HoursBetweenUpdates	numeric(5,2)	5

SQL Script

```
CREATE Proc [Process].[UspUpdate_PurchaseOrderInvoiceMapping]
    (
      @PrevCheck Int
    , @HoursBetweenUpdates Numeric(5 , 2)
    )
As
    Begin

--remove nocount on to speed up query

        Set NoCount On;

--check if table exists and create if it doesn't

        If ( Not Exists ( Select    1
                          From      [INFORMATION_SCHEMA].[TABLES]
                          Where     [TABLE_SCHEMA] = 'Lookups'
                                    And [TABLE_NAME] = 'PurchaseOrderInvoiceMapping'
)
           )
            Begin
                Create Table [Lookups].[PurchaseOrderInvoiceMapping]
                    (
                      [Company] Varchar(150) Collate Latin1_General_BIN
                    , [Grn] Varchar(20) Collate Latin1_General_BIN
                    , [Invoice] Varchar(20) Collate Latin1_General_BIN
                    , [PurchaseOrder] Varchar(20) Collate Latin1_General_BIN
                    , [LastUpdated] DateTime2
                    );
            End;


--check last time run and update if it's been longer than @HoursBetweenUpdates hours

        Declare @LastDate DateTime2;

        Select  @LastDate = Max([LastUpdated])
        From    [Lookups].[PurchaseOrderInvoiceMapping];
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Process.UspUpdate_-PurchaseOrderInvoiceMapping

```
        If @LastDate Is Null
            Or DateDiff(Minute , @LastDate , GetDate()) > ( @HoursBetweenUpdates
                                                            * 60 )
            Begin
    --Set time of run

                Declare @LastUpdated DateTime2;
                Select  @LastUpdated = GetDate();

--list the tables that are to be pulled back from each DB - if they are not found
the script will not be run against that db

                Declare @ListOfTables Varchar(Max) = 'GrnMatching,GrnDetails';

--create temporary tables to be pulled from different databases, including a column
to id

                Create Table [#Table1POIM]
                    (
                      [Company] Varchar(150) Collate Latin1_General_BIN
                    , [Grn] Varchar(20) Collate Latin1_General_BIN
                    , [Invoice] Varchar(20) Collate Latin1_General_BIN
                    , [PurchaseOrder] Varchar(20) Collate Latin1_General_BIN
                    );



--create script to pull data from each db into the tables

                Declare @Company Varchar(30) = 'All';
                Declare @SQL Varchar(Max) = 'USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
                    + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables
                    + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'',''))
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
                    Insert #Table1POIM
                            ( Company
                            , Grn
                            , Invoice
                            , PurchaseOrder
                            )
                    Select Distinct
                                Company = @DBCode
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Process.UspUpdate_-PurchaseOrderInvoiceMapping

```
                                , Grn = COALESCE(GM.Grn, GD.Grn)
                                , GM.Invoice
                                , PurchaseOrder = COALESCE(GD.PurchaseOrder,
PH.PurchaseOrder)
                            From
                                dbo.GrnMatching GM
                            Left Join dbo.GrnDetails GD
                                On GD.Grn = GM.Grn
                            Full Outer Join dbo.PorMasterHdr PH
                                On PH.PurchaseOrder = GD.PurchaseOrder;
            End
    End';

--execute script against each db, populating the base tables

                Exec [Process].[ExecForEachDB] @cmd = @SQL;

                Insert  [Lookups].[PurchaseOrderInvoiceMapping]
                        ( [Company]
                        , [Grn]
                        , [Invoice]
                        , [PurchaseOrder]
                        , [LastUpdated]
                        )
                        Select  [Company]
                              , [Grn]
                              , [Invoice]
                              , [PurchaseOrder]
                              , @LastUpdated
                        From    [#Table1POIM];

                If @PrevCheck = 1
                    Begin
                        Declare @CurrentCount Int
                          , @PreviousCount Int;

                        Select  @CurrentCount = Count(*)
                        From    [Lookups].[PurchaseOrderInvoiceMapping]
                        Where   [LastUpdated] = @LastUpdated;

                        Select  @PreviousCount = Count(*)
                        From    [Lookups].[PurchaseOrderInvoiceMapping]
                        Where   [LastUpdated] <> @LastUpdated;

                        If @PreviousCount > @CurrentCount
                            Begin
                                Delete  [Lookups].[PurchaseOrderInvoiceMapping]
                                Where   [LastUpdated] = @LastUpdated;
                                Print 'UspUpdate_PurchaseOrderInvoiceMapping - Count
has gone down since last run, no update applied';
                                Print 'Current Count = '
                                    + Cast(@CurrentCount As Varchar(5))
                                    + ' Previous Count = '
                                    + Cast(@PreviousCount As Varchar(5));
                            End;
                        If @PreviousCount <= @CurrentCount
                            Begin
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Process.UspUpdate_-PurchaseOrderInvoiceMapping

```
                                Delete  [Lookups].[PurchaseOrderInvoiceMapping]
                                Where   [LastUpdated] <> @LastUpdated;
                                Print 'UspUpdate_PurchaseOrderInvoiceMapping -
Update applied successfully';
                            End;
                    End;
                If @PrevCheck = 0
                    Begin
                        Delete  [Lookups].[PurchaseOrderInvoiceMapping]
                        Where   [LastUpdated] <> @LastUpdated;
                        Print 'UspUpdate_PurchaseOrderInvoiceMapping - Update
applied successfully';
                    End;
            End;
    End;
    If DateDiff(Minute , @LastDate , GetDate()) <= ( @HoursBetweenUpdates * 60 )
        Begin
            Print 'UspUpdate_PurchaseOrderInvoiceMapping - Table was last updated at
'
                + Cast(@LastDate As Varchar(255)) + ' no update applied';
        End;


GO
EXEC sp_addextendedproperty N'MS_Description', N'Stored proc to update specified
table', 'SCHEMA', N'Process', 'PROCEDURE', N'UspUpdate_PurchaseOrderInvoiceMapping',
NULL, NULL
GO
```

Uses

[Lookups].[PurchaseOrderInvoiceMapping]
[Process].[ExecForEachDB]
[Process]

Author: Johnson, Chris

[Process].[UspUpdate_PurchaseOrderStatus]

MS_Description

Stored proc to update specified table

Parameters

Name	Data Type	Max Length (Bytes)
@PrevCheck	int	4
@HoursBetweenUpdates	numeric(5,2)	5

SQL Script

```
CREATE Proc [Process].[UspUpdate_PurchaseOrderStatus]
    (
      @PrevCheck Int --if count is less than previous don't update

    , @HoursBetweenUpdates Numeric(5,2)
    )
As
    Begin
/*
Stored procedure created by Chris Johnson, Prometic Group September 2015 to populate
table with amounts relating to    Purchase Order Status details
*/

        Set NoCount On;

--check if table exists and create if it doesn't

        If ( Not Exists ( Select    1
                          From      [INFORMATION_SCHEMA].[TABLES]
                          Where     [TABLE_SCHEMA] = 'Lookups'
                                    And [TABLE_NAME] = 'PurchaseOrderStatus' )
           )
            Begin
                Create Table [Lookups].[PurchaseOrderStatus]
                    (
                      [Company] Varchar(150)                collate
latin1_general_bin
                    , [OrderStatusCode] Char(5)                collate
latin1_general_bin
                    , [OrderStatusDescription] Varchar(150)    collate
latin1_general_bin
                    , [LastUpdated] DateTime2
                    );
            End;


--check last time run and update if it's been longer than @HoursBetweenUpdates hours

        Declare @LastDate DateTime2;
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Process.UspUpdate_-PurchaseOrderStatus

```
        Select  @LastDate = Max([LastUpdated])
        From    [Lookups].[PurchaseOrderStatus];

        If @LastDate Is Null
            Or DateDiff(Minute , @LastDate , GetDate()) > (@HoursBetweenUpdates*60)
            Begin
    --Set time of run

                Declare @LastUpdated DateTime2;
                    Select  @LastUpdated = GetDate();

    --create master list of how codes affect stock

                Create Table [#OrdersPOStatus]
                    (
                      [OrderStatusCode] Varchar(5)            collate
latin1_general_bin
                    , [OrderStatusDescription] Varchar(150)    collate
latin1_general_bin
                    );

                Insert  [#OrdersPOStatus]
                        ( [OrderStatusCode]
                        , [OrderStatusDescription]
                        )
                        Select  [t].[OrderStatusCode]
                              , [t].[OrderStatusDescription]
                        From    ( Select    [OrderStatusCode] = '0'
                                          , [OrderStatusDescription] = 'In process'
                                  Union
                                  Select    [OrderStatusCode] = '1'
                                          , [OrderStatusDescription] = 'Ready to
print'
                                  Union
                                  Select    [OrderStatusCode] = '4'
                                          , [OrderStatusDescription] = 'Order
printed'
                                  Union
                                  Select    [OrderStatusCode] = '9'
                                          , [OrderStatusDescription] = 'Completed'
                                  Union
                                  Select    [OrderStatusCode] = '*'
                                          , [OrderStatusDescription] = 'Cancelled'
                                ) [t];

    --Get list of all companies in use


    --create temporary tables to be pulled from different databases, including a
column to id

                Create Table [#Table1]
                    (
                      [CompanyName] Varchar(150)    collate latin1_general_bin
                    );

    --create script to pull data from each db into the tables
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Process.UspUpdate_-PurchaseOrderStatus

```
                Declare @SQL Varchar(Max) = '
        USE [?];
        Declare @DB varchar(150),@DBCode varchar(150)
        Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end'
                    + --Only query DBs beginning SysProCompany

                    '
        IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
        BEGIN
        Insert #Table1
            ( CompanyName )
        Select @DBCode
        End';

    --execute script against each db, populating the base tables

                Exec [Process].[ExecForEachDB] @cmd = @SQL;

    --all companies process the same way

                Select  [T].[CompanyName]
                      , [O].[OrderStatusCode]
                      , [O].[OrderStatusDescription]
                Into    [#ResultsPoStatus]
                From    [#Table1] [T]
                        Left Join [#OrdersPOStatus] [O] On 1 = 1;

    --placeholder for anomalous results that are different to master list


                Insert  [Lookups].[PurchaseOrderStatus]
                        ( [Company]
                        , [OrderStatusCode]
                        , [OrderStatusDescription]
                        , [LastUpdated]
                        )
                        Select  [CompanyName]
                              , [OrderStatusCode]
                              , [OrderStatusDescription]
                              , @LastUpdated
                        From    [#ResultsPoStatus];

                If @PrevCheck = 1
                    Begin
                        Declare @CurrentCount Int
                          , @PreviousCount Int;

                        Select  @CurrentCount = Count(*)
                        From    [Lookups].[PurchaseOrderStatus]
                        Where   [LastUpdated] = @LastUpdated;

                        Select  @PreviousCount = Count(*)
                        From    [Lookups].[PurchaseOrderStatus]
                        Where   [LastUpdated] <> @LastUpdated;
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Process.UspUpdate_-PurchaseOrderStatus

```
                        If @PreviousCount > @CurrentCount
                            Begin
                                Delete  [Lookups].[PurchaseOrderStatus]
                                Where   [LastUpdated] = @LastUpdated;
                                Print 'UspUpdate_PurchaseOrderStatus - Count has
gone down since last run, no update applied';
                                Print 'Current Count = '
                                    + Cast(@CurrentCount As Varchar(5))
                                    + ' Previous Count = '
                                    + Cast(@PreviousCount As Varchar(5));
                            End;
                        If @PreviousCount <= @CurrentCount
                            Begin
                                Delete  [Lookups].[PurchaseOrderStatus]
                                Where   [LastUpdated] <> @LastUpdated;
                                Print 'UspUpdate_PurchaseOrderStatus - Update
applied successfully';
                            End;
                    End;
                If @PrevCheck = 0
                    Begin
                        Delete  [Lookups].[PurchaseOrderStatus]
                        Where   [LastUpdated] <> @LastUpdated;
                        Print 'UspUpdate_PurchaseOrderStatus - Update applied
successfully';
                    End;
            End;
    End;
    If DateDiff(Minute , @LastDate , GetDate()) <= (@HoursBetweenUpdates*60)
        Begin
            Print 'UspUpdate_PurchaseOrderStatus - Table was last updated at '
                + Cast(@LastDate As Varchar(255)) + ' no update applied';
        End;
GO
EXEC sp_addextendedproperty N'MS_Description', N'Stored proc to update specified
table', 'SCHEMA', N'Process', 'PROCEDURE', N'UspUpdate_PurchaseOrderStatus', NULL,
NULL
GO
```

Uses

[Lookups].[PurchaseOrderStatus]
[Process].[ExecForEachDB]
[Process]

Author: Johnson, Chris

[Process].[UspUpdate_PurchaseOrderTaxStatus]

MS_Description

Stored proc to update specified table

Parameters

Name	Data Type	Max Length (Bytes)
@PrevCheck	int	4
@HoursBetweenUpdates	numeric(5,2)	5

SQL Script

```
CREATE Proc [Process].[UspUpdate_PurchaseOrderTaxStatus]
    (
      @PrevCheck Int --if count is less than previous don't update

    , @HoursBetweenUpdates Numeric(5 , 2)
    )
As
    Begin
        Set NoCount On;


--check if table exists and create if it doesn't

        If ( Not Exists ( Select    1
                          From      [INFORMATION_SCHEMA].[TABLES]
                          Where     [TABLE_SCHEMA] = 'Lookups'
                                    And [TABLE_NAME] = 'PurchaseOrderTaxStatus' )
           )
            Begin
                Create Table [Lookups].[PurchaseOrderTaxStatus]
                    (
                      [Company] Varchar(150)
                    , [TaxStatusCode] Char(5)
                    , [TaxStatusDescription] Varchar(150)
                    , [LastUpdated] DateTime2
                    );
            End;


--check last time run and update if it's been longer than @HoursBetweenUpdates hours

        Declare @LastDate DateTime2;

        Select  @LastDate = Max([LastUpdated])
        From    [Lookups].[PurchaseOrderTaxStatus];

        If @LastDate Is Null
            Or DateDiff(Minute , @LastDate , GetDate()) > ( @HoursBetweenUpdates
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Process.UspUpdate_-PurchaseOrderTaxStatus

```
                                                        * 60 )
            Begin
    --Set time of run

                Declare @LastUpdated DateTime2;
                Select  @LastUpdated = GetDate();

    --create master list of how codes affect stock

                Create Table [#OrdersPOTax]
                    (
                      [TaxStatusCode] Varchar(5) Collate Latin1_General_BIN
                    , [TaxStatusDescription] Varchar(150) Collate Latin1_General_BIN
                    );

                Insert  [#OrdersPOTax]
                        ( [TaxStatusCode]
                        , [TaxStatusDescription]
                        )
                        Select  [t].[TaxStatusCode]
                              , [t].[TaxStatusDescription]
                        From    ( Select    [TaxStatusCode] = 'E'
                                          , [TaxStatusDescription] = 'Exempt from
tax'
                                  Union
                                  Select    [TaxStatusCode] = 'N'
                                          , [TaxStatusDescription] = 'Taxable'
                                ) [t];

    --Get list of all companies in use


    --create temporary tables to be pulled from different databases, including a
column to id

                Create Table [#Table1POTax]
                    (
                      [CompanyName] Varchar(150)
                    );

    --create script to pull data from each db into the tables

                Declare @SQL Varchar(Max) = 'USE [?];
        Declare @DB varchar(150),@DBCode varchar(150)
        Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end
        IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
        BEGIN
        Insert #Table1POTax
            ( CompanyName )
        Select @DBCode
        End';

    --execute script against each db, populating the base tables

                Exec [Process].[ExecForEachDB] @cmd = @SQL;

    --all companies process the same way
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Process.UspUpdate_-PurchaseOrderTaxStatus

```
                Select  [T].[CompanyName]
                      , [O].[TaxStatusCode]
                      , [O].[TaxStatusDescription]
                Into    [#ResultsPOTaxStatus]
                From    [#Table1POTax] [T]
                        Left Join [#OrdersPOTax] [O]
                            On 1 = 1;

                Insert  [Lookups].[PurchaseOrderTaxStatus]
                        ( [Company]
                        , [TaxStatusCode]
                        , [TaxStatusDescription]
                        , [LastUpdated]
                        )
                        Select  [CompanyName]
                              , [TaxStatusCode]
                              , [TaxStatusDescription]
                              , @LastUpdated
                        From    [#ResultsPOTaxStatus];

                If @PrevCheck = 1
                    Begin
                        Declare @CurrentCount Int
                          , @PreviousCount Int;

                        Select  @CurrentCount = Count(*)
                        From    [Lookups].[PurchaseOrderTaxStatus]
                        Where   [LastUpdated] = @LastUpdated;

                        Select  @PreviousCount = Count(*)
                        From    [Lookups].[PurchaseOrderTaxStatus]
                        Where   [LastUpdated] <> @LastUpdated;

                        If @PreviousCount > @CurrentCount
                            Begin
                                Delete  [Lookups].[PurchaseOrderTaxStatus]
                                Where   [LastUpdated] = @LastUpdated;
                                Print 'UspUpdate_PurchaseOrderTaxStatus - Count has
gone down since last run, no update applied';
                                Print 'Current Count = '
                                    + Cast(@CurrentCount As Varchar(5))
                                    + ' Previous Count = '
                                    + Cast(@PreviousCount As Varchar(5));
                            End;
                        If @PreviousCount <= @CurrentCount
                            Begin
                                Delete  [Lookups].[PurchaseOrderTaxStatus]
                                Where   [LastUpdated] <> @LastUpdated;
                                Print 'UspUpdate_PurchaseOrderTaxStatus - Update
applied successfully';
                            End;
                    End;
                If @PrevCheck = 0
                    Begin
                        Delete  [Lookups].[PurchaseOrderTaxStatus]
                        Where   [LastUpdated] <> @LastUpdated;
                        Print 'UspUpdate_PurchaseOrderTaxStatus - Update applied
```

Author: Johnson, Chris

```
successfully';
                    End;
            End;
    End;
    If DateDiff(Minute , @LastDate , GetDate()) <= ( @HoursBetweenUpdates * 60 )
        Begin
            Print 'UspUpdate_PurchaseOrderTaxStatus - Table was last updated at '
                + Cast(@LastDate As Varchar(255)) + ' no update applied';
        End;
GO
EXEC sp_addextendedproperty N'MS_Description', N'Stored proc to update specified
table', 'SCHEMA', N'Process', 'PROCEDURE', N'UspUpdate_PurchaseOrderTaxStatus',
NULL, NULL
GO
```

Uses

[Lookups].[PurchaseOrderTaxStatus]
[Process].[ExecForEachDB]
[Process]

Author: Johnson, Chris

[Process].[UspUpdate_PurchaseOrderType]

MS_Description

Stored proc to update specified table

Parameters

Name	Data Type	Max Length (Bytes)
@PrevCheck	int	4
@HoursBetweenUpdates	numeric(5,2)	5

SQL Script

```
CREATE Proc [Process].[UspUpdate_PurchaseOrderType]
    (
      @PrevCheck Int --if count is less than previous don't update

    , @HoursBetweenUpdates Numeric(5 , 2)
    )
As
    Begin

        Set NoCount On;


--check if table exists and create if it doesn't

        If ( Not Exists ( Select    1
                          From      [INFORMATION_SCHEMA].[TABLES]
                          Where     [TABLE_SCHEMA] = 'Lookups'
                                    And [TABLE_NAME] = 'PurchaseOrderType' )
           )
            Begin
                Create Table [Lookups].[PurchaseOrderType]
                    (
                      [Company] Varchar(150) Collate Latin1_General_BIN
                    , [OrderTypeCode] Char(5) Collate Latin1_General_BIN
                    , [OrderTypeDescription] Varchar(150) Collate Latin1_General_BIN
                    , [LastUpdated] DateTime2
                    );
            End;


--check last time run and update if it's been longer than @HoursBetweenUpdates hours

        Declare @LastDate DateTime2;

        Select  @LastDate = Max([LastUpdated])
        From    [Lookups].[PurchaseOrderType];

        If @LastDate Is Null
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Process.UspUpdate_-PurchaseOrderType

```
            Or DateDiff(Minute , @LastDate , GetDate()) > (@HoursBetweenUpdates*60)
            Begin
    --Set time of run

                Declare @LastUpdated DateTime2;
                Select  @LastUpdated = GetDate();

    --create master list of how codes affect stock

                Create Table [#OrdersPOType]
                    (
                      [OrderTypeCode] Varchar(5)
                    , [OrderTypeDescription] Varchar(150)
                    );

                Insert  [#OrdersPOType]
                        ( [OrderTypeCode]
                        , [OrderTypeDescription]
                        )
                        Select  [t].[OrderTypeCode]
                              , [t].[OrderTypeDescription]
                        From    ( Select    [OrderTypeCode] = 'L'
                                          , [OrderTypeDescription] = 'Local'
                                  Union
                                  Select    [OrderTypeCode] = 'I'
                                          , [OrderTypeDescription] = 'Import'
                                  Union
                                  Select    [OrderTypeCode] = 'O'
                                          , [OrderTypeDescription] = 'Other'
                                ) [t];

    --Get list of all companies in use


    --create temporary tables to be pulled from different databases, including a
column to id

                Create Table [#Table1POType]
                    (
                      [CompanyName] Varchar(150)
                    );

    --create script to pull data from each db into the tables

                Declare @SQL Varchar(Max) = 'USE [?];
        Declare @DB varchar(150),@DBCode varchar(150)
        Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end
        IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
        BEGIN
        Insert #Table1POType
            ( CompanyName )
        Select @DBCode
        End';

    --execute script against each db, populating the base tables
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Process.UspUpdate_-PurchaseOrderType

```
                Exec [Process].[ExecForEachDB] @cmd = @SQL;

    --all companies process the same way

                Select  [T].[CompanyName]
                      , [O].[OrderTypeCode]
                      , [O].[OrderTypeDescription]
                Into    [#ResultsPOType]
                From    [#Table1POType] [T]
                        Left Join [#OrdersPOType] [O]
                            On 1 = 1;

    --placeholder for anomalous results that are different to master list


                Insert  [Lookups].[PurchaseOrderType]
                        ( [Company]
                        , [OrderTypeCode]
                        , [OrderTypeDescription]
                        , [LastUpdated]
                        )
                        Select  [CompanyName]
                              , [OrderTypeCode]
                              , [OrderTypeDescription]
                              , @LastUpdated
                        From    [#ResultsPOType];

                If @PrevCheck = 1
                    Begin
                        Declare @CurrentCount Int
                          , @PreviousCount Int;

                        Select  @CurrentCount = Count(*)
                        From    [Lookups].[PurchaseOrderType]
                        Where   [LastUpdated] = @LastUpdated;

                        Select  @PreviousCount = Count(*)
                        From    [Lookups].[PurchaseOrderType]
                        Where   [LastUpdated] <> @LastUpdated;

                        If @PreviousCount > @CurrentCount
                            Begin
                                Delete  [Lookups].[PurchaseOrderType]
                                Where   [LastUpdated] = @LastUpdated;
                                Print 'UspUpdate_PurchaseOrderType - Count has gone
down since last run, no update applied';
                                Print 'Current Count = '
                                    + Cast(@CurrentCount As Varchar(5))
                                    + ' Previous Count = '
                                    + Cast(@PreviousCount As Varchar(5));
                            End;
                        If @PreviousCount <= @CurrentCount
                            Begin
                                Delete  [Lookups].[PurchaseOrderType]
                                Where   [LastUpdated] <> @LastUpdated;
                                Print 'UspUpdate_PurchaseOrderType - Update applied
successfully';
```

Author: Johnson, Chris

```
                            End;
                    End;
                If @PrevCheck = 0
                    Begin
                        Delete  [Lookups].[PurchaseOrderType]
                        Where   [LastUpdated] <> @LastUpdated;
                        Print 'UspUpdate_PurchaseOrderType - Update applied
successfully';
                    End;
            End;
    End;
    If DateDiff(Minute , @LastDate , GetDate()) <= (@HoursBetweenUpdates*60)
        Begin
            Print 'UspUpdate_PurchaseOrderType - Table was last updated at '
                + Cast(@LastDate As Varchar(255)) + ' no update applied';
        End;
GO
EXEC sp_addextendedproperty N'MS_Description', N'Stored proc to update specified
table', 'SCHEMA', N'Process', 'PROCEDURE', N'UspUpdate_PurchaseOrderType', NULL,
NULL
GO
```

Uses

[Lookups].[PurchaseOrderType]
[Process].[ExecForEachDB]
[Process]

[Process].[UspUpdate_ReqBuyers]

MS_Description

Stored proc to update specified table

Parameters

Name	Data Type	Max Length (Bytes)
@PrevCheck	int	4
@HoursBetweenUpdates	numeric(5,2)	5

SQL Script

```
CREATE Proc [Process].[UspUpdate_ReqBuyers]
    (
      @PrevCheck Int
    , @HoursBetweenUpdates Numeric(5 , 2)
    )
As
    Begin
--remove nocount on to speed up query

        Set NoCount On;

--check if table exists and create if it doesn't

        If ( Not Exists ( Select    1
                          From      [INFORMATION_SCHEMA].[TABLES]
                          Where     [TABLE_SCHEMA] = 'Lookups'
                                    And [TABLE_NAME] = 'ReqBuyers' )
           )
            Begin
                Create Table [Lookups].[ReqBuyers]
                    (
                      [BuyerName] Varchar(150)
                    , [LastUpdated] DateTime2
                    );
            End;



--check last time run and update if it's been longer than @HoursBetweenUpdates hours

        Declare @LastDate DateTime2;

        Select  @LastDate = Max([LastUpdated])
        From    [Lookups].[ReqBuyers];

        If @LastDate Is Null
            Or DateDiff(Minute , @LastDate , GetDate()) > ( @HoursBetweenUpdates
                                                            * 60 )
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Process.UspUpdate_Req-Buyers

```
            Begin
    --Set time of run

                Declare @LastUpdated DateTime2;
                Select  @LastUpdated = GetDate();

--list the tables that are to be pulled back from each DB - if they are not found
the script will not be run against that db

                Declare @ListOfTables Varchar(Max) = 'InvBuyer';

--create temporary tables to be pulled from different databases, including a column
to id

                Create Table [#Table1]
                    (
                      [Company] Varchar(150) Collate Latin1_General_BIN
                    , [BuyerName] Varchar(150) Collate Latin1_General_BIN
                    );

--create script to pull data from each db into the tables

                Declare @Company Varchar(30) = 'All';
                Declare @SQL Varchar(Max) = 'USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN
        IF isnumeric(@DBCode) = 1
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables
                    + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_Bin From Black-
Box.dbo.udf_SplitString(@ListOfTables,'',''))
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
                Insert #Table1
                    ( Company, [BuyerName])
                Select Distinct @DBCode
                ,[Buyer] = Case When [RD].[Buyer] = '''' Then ''Blank''
                When [RD].[Buyer] = '' '' Then ''Blank''
                       Else [RD].[Buyer]
                  End
From    [dbo].[ReqDetail] As [RD];
            End
    End';


--execute script against each db, populating the base tables

                Exec [Process].[ExecForEachDB] @cmd = @SQL;

                Insert  [Lookups].[ReqBuyers]
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Process.UspUpdate_Req-Buyers

```
                        ( [BuyerName]
                        , [LastUpdated]
                        )
                        Select  Distinct
                                [BuyerName]
                              , @LastUpdated
                        From    [#Table1];

                If @PrevCheck = 1
                    Begin
                        Declare @CurrentCount Int
                          , @PreviousCount Int;

                        Select  @CurrentCount = Count(*)
                        From    [Lookups].[ReqBuyers]
                        Where   [LastUpdated] = @LastUpdated;

                        Select  @PreviousCount = Count(*)
                        From    [Lookups].[ReqBuyers]
                        Where   [LastUpdated] <> @LastUpdated;

                        If @PreviousCount > @CurrentCount
                            Begin
                                Delete  [Lookups].[ReqBuyers]
                                Where   [LastUpdated] = @LastUpdated;
                                Print 'UspUpdate_ReqBuyers - Count has gone down
since last run, no update applied';
                                Print 'Current Count = '
                                    + Cast(@CurrentCount As Varchar(5))
                                    + ' Previous Count = '
                                    + Cast(@PreviousCount As Varchar(5));
                            End;
                        If @PreviousCount <= @CurrentCount
                            Begin
                                Delete  [Lookups].[ReqBuyers]
                                Where   [LastUpdated] <> @LastUpdated;
                                Print 'UspUpdate_ReqBuyers - Update applied
successfully';
                            End;
                    End;
                If @PrevCheck = 0
                    Begin
                        Delete  [Lookups].[ReqBuyers]
                        Where   [LastUpdated] <> @LastUpdated;
                        Print 'UspUpdate_ReqBuyers - Update applied successfully';
                    End;
            End;
    End;
    If DateDiff(Minute , @LastDate , GetDate()) <= ( @HoursBetweenUpdates * 60 )
        Begin
            Print 'UspUpdate_ReqBuyers - Table was last updated at '
                + Cast(@LastDate As Varchar(255)) + ' no update applied';
        End;

GO
EXEC sp_addextendedproperty N'MS_Description', N'Stored proc to update specified
table', 'SCHEMA', N'Process', 'PROCEDURE', N'UspUpdate_ReqBuyers', NULL, NULL
GO
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Process.UspUpdate_Req-Buyers

Uses

[Lookups].[ReqBuyers]

[Process].[ExecForEachDB]

[Process]

Author: Johnson, Chris

[Process].[UspUpdate_ReqnStatus]

MS_Description

Stored proc to update specified table

Parameters

Name	Data Type	Max Length (Bytes)
@PrevCheck	int	4
@HoursBetweenUpdates	numeric(5,2)	5

SQL Script

```
CREATE Proc [Process].[UspUpdate_ReqnStatus]
    (
      @PrevCheck Int --if count is less than previous don't update

    , @HoursBetweenUpdates Numeric(5 , 2)
    )
As
    Begin
        Set NoCount On;

--check if table exists and create if it doesn't

        If ( Not Exists ( Select    1
                          From      [INFORMATION_SCHEMA].[TABLES]
                          Where     [TABLE_SCHEMA] = 'Lookups'
                                    And [TABLE_NAME] = 'ReqnStatus' )
           )
            Begin
                Create Table [Lookups].[ReqnStatus]
                    (
                      [Company] Varchar(150)                collate
latin1_general_bin
                    , [ReqnStatusCode] Char(5)                collate
latin1_general_bin
                    , [ReqnStatusDescription] Varchar(150)    collate
latin1_general_bin
                    , [LastUpdated] DateTime2
                    );
            End;


--check last time run and update if it's been longer than @HoursBetweenUpdates hours

        Declare @LastDate DateTime2;

        Select  @LastDate = Max([LastUpdated])
        From    [Lookups].[ReqnStatus];
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Process.UspUpdate_-ReqnStatus

```
        If @LastDate Is Null
            Or DateDiff(Minute , @LastDate , GetDate()) > ( @HoursBetweenUpdates
                                                            * 60 )
            Begin
    --Set time of run

                Declare @LastUpdated DateTime2;
                Select  @LastUpdated = GetDate();

    --create master list of how codes affect stock

                Create Table [#OrdersReqSC]
                    (
                      [ReqnStatusCode] Varchar(5)            collate
latin1_general_bin
                    , [ReqnStatusDescription] Varchar(150)    collate
latin1_general_bin
                    );

                Insert  [#OrdersReqSC]
                        ( [ReqnStatusCode]
                        , [ReqnStatusDescription]
                        )
                        Select  [t].[ReqnStatusCode]
                              , [t].[ReqnStatusDescription]
                        From    ( Select    [ReqnStatusCode] = ''
                                          , [ReqnStatusDescription] = 'Normal'
                                  Union
                                  Select    [ReqnStatusCode] = 'R'
                                          , [ReqnStatusDescription] = 'Approved and
Ready'
                                  Union
                                  Select    [ReqnStatusCode] = 'P'
                                          , [ReqnStatusDescription] = 'Confirmed
into a Purchase Order'
                                  Union
                                  Select    [ReqnStatusCode] = 'I'
                                          , [ReqnStatusDescription] = 'Issued'
                                  Union
                                  Select    [ReqnStatusCode] = 'T'
                                          , [ReqnStatusDescription] = 'Transferred'
                                  Union
                                  Select    [ReqnStatusCode] = '*'
                                          , [ReqnStatusDescription] = 'Cancelled'
                                ) [t];

    --Get list of all companies in use


    --create temporary tables to be pulled from different databases, including a
column to id

                Create Table [#Table1ReqSC]
                    (
                      [CompanyName] Varchar(150)
                    );

    --create script to pull data from each db into the tables
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Process.UspUpdate_-
ReqnStatus

```
                Declare @SQL Varchar(Max) = '
        USE [?];
        Declare @DB varchar(150),@DBCode varchar(150)
        Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end'
                    + --Only query DBs beginning SysProCompany

                    '
        IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
        BEGIN
        Insert #Table1ReqSC
            ( CompanyName )
        Select @DBCode
        End';

    --execute script against each db, populating the base tables

                Exec [Process].[ExecForEachDB] @cmd = @SQL;

    --all companies process the same way

                Select  [T].[CompanyName]
                      , [O].[ReqnStatusCode]
                      , [O].[ReqnStatusDescription]
                Into    [#ResultsReqStatus]
                From    [#Table1ReqSC] [T]
                        Left Join [#OrdersReqSC] [O]
                            On 1 = 1;

    --placeholder for anomalous results that are different to master list

                Insert  [Lookups].[ReqnStatus]
                        ( [Company]
                        , [ReqnStatusCode]
                        , [ReqnStatusDescription]
                        , [LastUpdated]
                        )
                        Select  [CompanyName]
                              , [ReqnStatusCode]
                              , [ReqnStatusDescription]
                              , @LastUpdated
                        From    [#ResultsReqStatus];

                If @PrevCheck = 1
                    Begin
                        Declare @CurrentCount Int
                          , @PreviousCount Int;

                        Select  @CurrentCount = Count(*)
                        From    [Lookups].[ReqnStatus]
                        Where   [LastUpdated] = @LastUpdated;

                        Select  @PreviousCount = Count(*)
                        From    [Lookups].[ReqnStatus]
                        Where   [LastUpdated] <> @LastUpdated;
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Process.UspUpdate_-ReqnStatus

```
                        If @PreviousCount > @CurrentCount
                            Begin
                                Delete  [Lookups].[ReqnStatus]
                                Where   [LastUpdated] = @LastUpdated;
                                Print 'UspUpdate_ReqnStatus - Count has gone down
since last run, no update applied';
                                Print 'Current Count = '
                                    + Cast(@CurrentCount As Varchar(5))
                                    + ' Previous Count = '
                                    + Cast(@PreviousCount As Varchar(5));
                            End;
                        If @PreviousCount <= @CurrentCount
                            Begin
                                Delete  [Lookups].[ReqnStatus]
                                Where   [LastUpdated] <> @LastUpdated;
                                Print 'UspUpdate_ReqnStatus - Update applied
successfully';
                            End;
                    End;
                If @PrevCheck = 0
                    Begin
                        Delete  [Lookups].[ReqnStatus]
                        Where   [LastUpdated] <> @LastUpdated;
                        Print 'UspUpdate_ReqnStatus - Update applied successfully';
                    End;
            End;
    End;
    If DateDiff(Minute , @LastDate , GetDate()) <= ( @HoursBetweenUpdates * 60 )
        Begin
            Print 'UspUpdate_ReqnStatus - Table was last updated at '
                + Cast(@LastDate As Varchar(255)) + ' no update applied';
        End;
GO
EXEC sp_addextendedproperty N'MS_Description', N'Stored proc to update specified
table', 'SCHEMA', N'Process', 'PROCEDURE', N'UspUpdate_ReqnStatus', NULL, NULL
GO
```

Uses

[Lookups].[ReqnStatus]
[Process].[ExecForEachDB]
[Process]

Author: Johnson, Chris

[Process].[UspUpdate_SalesOrderDocumentType]

MS_Description

Stored proc to update specified table

Parameters

Name	Data Type	Max Length (Bytes)
@PrevCheck	int	4
@HoursBetweenUpdates	numeric(5,2)	5

SQL Script

```
CREATE Proc [Process].[UspUpdate_SalesOrderDocumentType]
    (
      @PrevCheck Int --if count is less than previous don't update

    , @HoursBetweenUpdates Numeric(5 , 2)
    )
As
    Begin

--check if table exists and create if it doesn't

        If ( Not Exists ( Select    1
                          From      [INFORMATION_SCHEMA].[TABLES]
                          Where     [TABLE_SCHEMA] = 'Lookups'
                                    And [TABLE_NAME] = 'SalesOrderDocumentType' )
           )
            Begin
                Create Table [Lookups].[SalesOrderDocumentType]
                    (
                      [DocumentType] Varchar(10) Collate Latin1_General_BIN
                    , [DocumentTypeDesc] Varchar(250) Collate Latin1_General_BIN
                    , [LastUpdated] DateTime2
                    );
            End;


--check last time run and update if it's been longer than @HoursBetweenUpdates hours

        Declare @LastDate DateTime2;

        Select  @LastDate = Max([LastUpdated])
        From    [Lookups].[SalesOrderDocumentType];

        If @LastDate Is Null
            Or DateDiff(Minute , @LastDate , GetDate()) > ( @HoursBetweenUpdates
                                                            * 60 )
            Begin
    --Set time of run
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Process.UspUpdate_-SalesOrderDocumentType

```
                Declare @LastUpdated DateTime2;
                Select  @LastUpdated = GetDate();
    --create master list of how codes affect stock

                Create Table [#SalesOrderDocumentType]
                    (
                      [DocumentType] Varchar(10) Collate Latin1_General_BIN
                    , [DocumentTypeDescription] Varchar(250) Collate Latin1_General_-
BIN
                    );

                Insert  [#SalesOrderDocumentType]
                        ( [DocumentType]
                        , [DocumentTypeDescription]
                        )
                        Select  [t].[DocumentType]
                              , [t].[DocumentTypeDescription]
                        From    ( Select    [DocumentType] = 'B'
                                          , [DocumentTypeDescription] = 'Billing'
                                  Union
                                  Select    [DocumentType] = 'O'
                                          , [DocumentTypeDescription] = 'Order'
                                  Union
                                  Select    [DocumentType] = 'C'
                                          , [DocumentTypeDescription] = 'Credit
Note'
                                  Union
                                  Select    [DocumentType] = 'D'
                                          , [DocumentTypeDescription] = 'Debit Note'
                                ) [t];



    --placeholder for anomalous results that are different to master list


                Insert  [Lookups].[SalesOrderDocumentType]
                        ( [DocumentType]
                        , [DocumentTypeDesc]
                        , [LastUpdated]
                        )
                        Select  [SODT].[DocumentType]
                              , [SODT].[DocumentTypeDescription]
                              , @LastUpdated
                        From    [#SalesOrderDocumentType] [SODT];

                If @PrevCheck = 1
                    Begin
                        Declare @CurrentCount Int
                          , @PreviousCount Int;

                        Select  @CurrentCount = Count(*)
                        From    [Lookups].[SalesOrderDocumentType] As [cn]
                        Where   [cn].[LastUpdated] = @LastUpdated;

                        Select  @PreviousCount = Count(*)
```

Author: Johnson, Chris

```
                        From    [Lookups].[SalesOrderDocumentType] As [cn]
                        Where   [cn].[LastUpdated] <> @LastDate;

                        If @PreviousCount > @CurrentCount
                            Begin
                                Delete  [Lookups].[SalesOrderDocumentType]
                                Where   [LastUpdated] = @LastDate;
                                Print 'UspUpdate_SalesOrderDocumentType - Count has
gone down since last run, no update applied';
                                Print 'Current Count = '
                                    + Cast(@CurrentCount As Varchar(5))
                                    + ' Previous Count = '
                                    + Cast(@PreviousCount As Varchar(5));
                            End;
                        If @PreviousCount <= @CurrentCount
                            Begin
                                Delete  [Lookups].[SalesOrderDocumentType]
                                Where   [LastUpdated] <> @LastUpdated
                                        Or [LastUpdated] Is Null;
                                Print 'UspUpdate_SalesOrderDocumentType - Update
applied successfully';
                            End;
                    End;
                If @PrevCheck = 0
                    Begin
                        Delete  [Lookups].[SalesOrderDocumentType]
                        Where   [LastUpdated] <> @LastUpdated;
                        Print 'UspUpdate_SalesOrderDocumentType - Update applied
successfully';
                    End;
            End;
    End;
    If DateDiff(Minute , @LastDate , GetDate()) <= ( @HoursBetweenUpdates * 60 )
        Begin
            Print 'UspUpdate_SalesOrderDocumentType - Table was last updated at '
                + Cast(@LastDate As Varchar(255)) + ' no update applied';
        End;
GO
EXEC sp_addextendedproperty N'MS_Description', N'Stored proc to update specified
table', 'SCHEMA', N'Process', 'PROCEDURE', N'UspUpdate_SalesOrderDocumentType',
NULL, NULL
GO
```

Uses

[Lookups].[SalesOrderDocumentType]
[Process]

Author: Johnson, Chris

[Process].[UspUpdate_SalesOrderLineType]

MS_Description

Stored proc to update specified table

Parameters

Name	Data Type	Max Length (Bytes)
@PrevCheck	int	4
@HoursBetweenUpdates	numeric(5,2)	5

SQL Script

```
CREATE Proc [Process].[UspUpdate_SalesOrderLineType]
    (
      @PrevCheck Int --if count is less than previous don't update

    , @HoursBetweenUpdates Numeric(5 , 2)
    )
As
    Begin
        Set NoCount On;

--check if table exists and create if it doesn't

        If ( Not Exists ( Select    1
                          From      [INFORMATION_SCHEMA].[TABLES]
                          Where     [TABLE_SCHEMA] = 'Lookups'
                                    And [TABLE_NAME] = 'SalesOrderLineType' )
           )
            Begin
                Create Table [Lookups].[SalesOrderLineType]
                    (
                      [Company] Varchar(150)                    collate
latin1_general_bin
                    , [LineTypeCode] Char(5)                    collate
latin1_general_bin
                    , [LineTypeDescription] Varchar(150)        collate
latin1_general_bin
                    , [LastUpdated] DateTime2
                    );
            End;


--check last time run and update if it's been longer than @HoursBetweenUpdates hours

        Declare @LastDate DateTime2;

        Select  @LastDate = Max([LastUpdated])
        From    [Lookups].[SalesOrderLineType];
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Process.UspUpdate_-SalesOrderLineType

```
        If @LastDate Is Null
            Or DateDiff(Minute , @LastDate , GetDate()) > ( @HoursBetweenUpdates
                                                            * 60 )
            Begin
    --Set time of run

                Declare @LastUpdated DateTime2;
                Select  @LastUpdated = GetDate();

    --create master list of how codes affect stock

                Create Table [#SalesOrderLineType]
                    (
                      [LineTypeCode] Varchar(5)                collate
latin1_general_bin
                    , [LineTypeDescription] Varchar(150)    collate
latin1_general_bin
                    );

                Insert  [#SalesOrderLineType]
                        ( [LineTypeCode]
                        , [LineTypeDescription]
                        )
                        Select  [t].[LineTypeCode]
                              , [t].[LineTypeDescription]
                        From    ( Select    [LineTypeCode] = '1'
                                          , [LineTypeDescription] = 'Stocked
Merchandise'
                                  Union
                                  Select    [LineTypeCode] = '4'
                                          , [LineTypeDescription] = 'Freight'
                                  Union
                                  Select    [LineTypeCode] = '5'
                                          , [LineTypeDescription] = 'Miscellaneous
Charges'
                                  Union
                                  Select    [LineTypeCode] = '6'
                                          , [LineTypeDescription] = 'Comment Line'
                                  Union
                                  Select    [LineTypeCode] = '7'
                                          , [LineTypeDescription] = 'Non-stocked
Merchandise'
                                ) [t];


    --create temporary tables to be pulled from different databases, including a
column to id

                Create Table [#SalesOrderLineTypeTable1]
                    (
                      [CompanyName] Varchar(150)
                    );

    --create script to pull data from each db into the tables

                Declare @SQL Varchar(Max) = 'USE [?];
        Declare @DB varchar(150),@DBCode varchar(150)
        Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Process.UspUpdate_-SalesOrderLineType

```
        IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
        BEGIN
        Insert #SalesOrderLineTypeTable1
            ( CompanyName )
        Select @DBCode
        End';

    --execute script against each db, populating the base tables

                Exec [Process].[ExecForEachDB] @cmd = @SQL;

    --all companies process the same way

                Select  [T].[CompanyName]
                      , [O].[LineTypeCode]
                      , [O].[LineTypeDescription]
                Into    [#ResultsPoStatus]
                From    [#SalesOrderLineTypeTable1] [T]
                        Cross Join ( Select [LineTypeCode]
                                            , [LineTypeDescription]
                                     From   [#SalesOrderLineType]
                                   ) [O];

    --placeholder for anomalous results that are different to master list


                Insert  [Lookups].[SalesOrderLineType]
                        ( [Company]
                        , [LineTypeCode]
                        , [LineTypeDescription]
                        , [LastUpdated]
                        )
                        Select  [CompanyName]
                              , [LineTypeCode]
                              , [LineTypeDescription]
                              , @LastUpdated
                        From    [#ResultsPoStatus];

                If @PrevCheck = 1
                    Begin
                        Declare @CurrentCount Int
                          , @PreviousCount Int;

                        Select  @CurrentCount = Count(*)
                        From    [Lookups].[SalesOrderLineType]
                        Where   [LastUpdated] = @LastUpdated;

                        Select  @PreviousCount = Count(*)
                        From    [Lookups].[SalesOrderLineType]
                        Where   [LastUpdated] <> @LastUpdated;

                        If @PreviousCount > @CurrentCount
                            Begin
                                Delete  [Lookups].[SalesOrderLineType]
                                Where   [LastUpdated] = @LastUpdated;
                                Print 'UspUpdate_SalesOrderLineType - Count has gone
down since last run, no update applied';
```

Author: Johnson, Chris

```
                                Print 'Current Count = '
                                    + Cast(@CurrentCount As Varchar(5))
                                    + ' Previous Count = '
                                    + Cast(@PreviousCount As Varchar(5));
                            End;
                        If @PreviousCount <= @CurrentCount
                            Begin
                                Delete  [Lookups].[SalesOrderLineType]
                                Where   [LastUpdated] <> @LastUpdated;
                                Print 'UspUpdate_SalesOrderLineType - Update applied
successfully';
                            End;
                    End;
                If @PrevCheck = 0
                    Begin
                        Delete  [Lookups].[SalesOrderLineType]
                        Where   [LastUpdated] <> @LastUpdated;
                        Print 'UspUpdate_SalesOrderLineType - Update applied
successfully';
                    End;
            End;
    End;
    If DateDiff(Minute , @LastDate , GetDate()) <= ( @HoursBetweenUpdates * 60 )
        Begin
            Print 'UspUpdate_SalesOrderLineType - Table was last updated at '
                + Cast(@LastDate As Varchar(255)) + ' no update applied';
        End;
GO
EXEC sp_addextendedproperty N'MS_Description', N'Stored proc to update specified
table', 'SCHEMA', N'Process', 'PROCEDURE', N'UspUpdate_SalesOrderLineType', NULL,
NULL
GO
```

Uses

[Lookups].[SalesOrderLineType]
[Process].[ExecForEachDB]
[Process]

[Process].[UspUpdate_SalesOrderStatus]

MS_Description

Stored proc to update specified table

Parameters

Name	Data Type	Max Length (Bytes)
@PrevCheck	int	4
@HoursBetweenUpdates	numeric(5,2)	5

SQL Script

```
CREATE Proc [Process].[UspUpdate_SalesOrderStatus]
    (
      @PrevCheck Int --if count is less than previous don't update

    , @HoursBetweenUpdates Numeric(5 , 2)
    )
As
    Begin

        Set NoCount On;

--check if table exists and create if it doesn't

        If ( Not Exists ( Select    1
                          From      [INFORMATION_SCHEMA].[TABLES]
                          Where     [TABLE_SCHEMA] = 'Lookups'
                                    And [TABLE_NAME] = 'SalesOrderStatus' )
           )
            Begin
                Create Table [Lookups].[SalesOrderStatus]
                    (
                      [Company] Varchar(150)                    collate
latin1_general_bin
                    , [OrderStatusCode] Char(5)                     collate
latin1_general_bin
                    , [OrderStatusDescription] Varchar(150)         collate
latin1_general_bin
                    , [LastUpdated] DateTime2
                    );
            End;


--check last time run and update if it's been longer than @HoursBetweenUpdates hours

        Declare @LastDate DateTime2;

        Select  @LastDate = Max([LastUpdated])
        From    [Lookups].[SalesOrderStatus];
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Process.UspUpdate_-SalesOrderStatus

```
        If @LastDate Is Null
            Or DateDiff(Minute , @LastDate , GetDate()) > ( @HoursBetweenUpdates
                                                            * 60 )
            Begin
    --Set time of run

                Declare @LastUpdated DateTime2;
                Select  @LastUpdated = GetDate();

    --create master list of how codes affect stock

                Create Table [#OrdersSalesStatus]
                    (
                      [OrderStatusCode] Varchar(5)
                    , [OrderStatusDescription] Varchar(150)
                    );

                Insert  [#OrdersSalesStatus]
                        ( [OrderStatusCode]
                        , [OrderStatusDescription]
                        )
                        Select  [t].[OrderStatusCode]
                              , [t].[OrderStatusDescription]
                        From    ( Select    [OrderStatusCode] = '0'
                                          , [OrderStatusDescription] = 'In process'
                                  Union
                                  Select    [OrderStatusCode] = '1'
                                          , [OrderStatusDescription] = 'Open Order'
                                  Union
                                  Select    [OrderStatusCode] = '2'
                                          , [OrderStatusDescription] = 'Open
Backorder'
                                  Union
                                  Select    [OrderStatusCode] = '3'
                                          , [OrderStatusDescription] = 'Released
Backorder'
                                  Union
                                  Select    [OrderStatusCode] = '4'
                                          , [OrderStatusDescription] = 'In
Warehouse'
                                  Union
                                  Select    [OrderStatusCode] = '8'
                                          , [OrderStatusDescription] = 'Ready to
Invoice'
                                  Union
                                  Select    [OrderStatusCode] = '9'
                                          , [OrderStatusDescription] = 'Complete'
                                  Union
                                  Select    [OrderStatusCode] = '*'
                                          , [OrderStatusDescription] = 'Cancelled
during entry'
                                  Union
                                  Select    [OrderStatusCode] = '\'
                                          , [OrderStatusDescription] = 'Cancelled'
                                  Union
                                  Select    [OrderStatusCode] = 'F'
                                          , [OrderStatusDescription] = 'Forward'
```

Author: Johnson, Chris

```
                                    Union
                                    Select    [OrderStatusCode] = 'S'
                                            , [OrderStatusDescription] = 'Suspense'
                                ) [t];

    --Get list of all companies in use


    --create temporary tables to be pulled from different databases, including a
column to id

                Create Table [#Table1]
                    (
                      [CompanyName] Varchar(150)
                    );

    --create script to pull data from each db into the tables

                Declare @SQL Varchar(Max) = '
        USE [?];
        Declare @DB varchar(150),@DBCode varchar(150)
        Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end'
                    + --Only query DBs beginning SysProCompany

                    '
        IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
        BEGIN
        Insert #Table1
            ( CompanyName )
        Select @DBCode
        End';

    --execute script against each db, populating the base tables

                Exec [Process].[ExecForEachDB] @cmd = @SQL;

    --all companies process the same way

                Select  [T].[CompanyName]
                      , [O].[OrderStatusCode]
                      , [O].[OrderStatusDescription]
                Into    [#ResultsPoStatus]
                From    [#Table1] [T]
                        Cross Join ( Select [OrderStatusCode]
                                           , [OrderStatusDescription]
                                     From   [#OrdersSalesStatus]
                                   ) [O];

    --placeholder for anomalous results that are different to master list


                Insert  [Lookups].[SalesOrderStatus]
                        ( [Company]
                        , [OrderStatusCode]
                        , [OrderStatusDescription]
                        , [LastUpdated]
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Process.UspUpdate_-SalesOrderStatus

```
                        )
                        Select  [CompanyName]
                              , [OrderStatusCode]
                              , [OrderStatusDescription]
                              , @LastUpdated
                        From    [#ResultsPoStatus];

                If @PrevCheck = 1
                    Begin
                        Declare @CurrentCount Int
                          , @PreviousCount Int;

                        Select  @CurrentCount = Count(*)
                        From    [Lookups].[SalesOrderStatus]
                        Where   [LastUpdated] = @LastUpdated;

                        Select  @PreviousCount = Count(*)
                        From    [Lookups].[SalesOrderStatus]
                        Where   [LastUpdated] <> @LastUpdated;

                        If @PreviousCount > @CurrentCount
                            Begin
                                Delete  [Lookups].[SalesOrderStatus]
                                Where   [LastUpdated] = @LastUpdated;
                                Print 'UspUpdate_SalesOrderStatus - Count has gone
down since last run, no update applied';
                                Print 'Current Count = '
                                    + Cast(@CurrentCount As Varchar(5))
                                    + ' Previous Count = '
                                    + Cast(@PreviousCount As Varchar(5));
                            End;
                        If @PreviousCount <= @CurrentCount
                            Begin
                                Delete  [Lookups].[SalesOrderStatus]
                                Where   [LastUpdated] <> @LastUpdated;
                                Print 'UspUpdate_SalesOrderStatus - Update applied
successfully';
                            End;
                    End;
                If @PrevCheck = 0
                    Begin
                        Delete  [Lookups].[SalesOrderStatus]
                        Where   [LastUpdated] <> @LastUpdated;
                        Print 'UspUpdate_SalesOrderStatus - Update applied
successfully';
                    End;
            End;
    End;
    If DateDiff(Minute , @LastDate , GetDate()) <= ( @HoursBetweenUpdates * 60 )
        Begin
            Print 'UspUpdate_SalesOrderStatus - Table was last updated at '
                + Cast(@LastDate As Varchar(255)) + ' no update applied';
        End;
GO
EXEC sp_addextendedproperty N'MS_Description', N'Stored proc to update specified
table', 'SCHEMA', N'Process', 'PROCEDURE', N'UspUpdate_SalesOrderStatus', NULL, NULL
GO
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Process.UspUpdate_-SalesOrderStatus

Uses

[Lookups].[SalesOrderStatus]

[Process].[ExecForEachDB]

[Process]

Author: Johnson, Chris

[Process].[UspUpdate_StockCode]

MS_Description

Stored proc to update specified table

Parameters

Name	Data Type	Max Length (Bytes)
@PrevCheck	int	4
@HoursBetweenUpdates	numeric(5,2)	5

SQL Script

```
CREATE Proc [Process].[UspUpdate_StockCode]
    (
      @PrevCheck Int
    , @HoursBetweenUpdates Numeric(5 , 2)
    )
As
    Begin
--remove nocount on to speed up query

        Set NoCount On;

--check if table exists and create if it doesn't

        If ( Not Exists ( Select    1
                          From      [INFORMATION_SCHEMA].[TABLES]
                          Where     [TABLE_SCHEMA] = 'Lookups'
                                    And [TABLE_NAME] = 'StockCode' )
           )
            Begin
                Create Table [Lookups].[StockCode]
                    (
                      [Company] Varchar(150)            collate Latin1_General_BIN
                    , [StockCode] Varchar(150)            collate Latin1_General_BIN
                    , [StockDescription] Varchar(150)    collate Latin1_General_BIN
                    , [PartCategory] Varchar(5)           collate Latin1_General_-
BIN
                    , [ActivePOFlag] Bit
                    , [LastUpdated] DateTime2
                    );
            End;

--check last time run and update if it's been longer than @HoursBetweenUpdates hours

        Declare @LastDate DateTime2;

        Select  @LastDate = Max([LastUpdated])
        From    [Lookups].[StockCode];
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Process.UspUpdate_-
StockCode

```
        If @LastDate Is Null
            Or DateDiff(Minute , @LastDate , GetDate()) > ( @HoursBetweenUpdates
                                                            * 60 )
            Begin
    --Set time of run

                Declare @LastUpdated DateTime2;
                Select  @LastUpdated = GetDate();

--list the tables that are to be pulled back from each DB - if they are not found
the script will not be run against that db

                Declare @ListOfTables Varchar(Max) = 'InvMaster';

--create temporary tables to be pulled from different databases, including a column
to id

                Create Table [#Table1StockCode]
                    (
                      [Company] Varchar(150)            collate Latin1_General_BIN
                    , [StockCode] Varchar(150)            collate Latin1_General_BIN
                    , [StockDescription] Varchar(150)    collate Latin1_General_BIN
                    , [PartCategory] Varchar(5)            collate Latin1_General_-
BIN
                    , [ActivePOFlag] Bit
                    );

--create script to pull data from each db into the tables

                Declare @Company Varchar(30) = 'All';
                Declare @SQL Varchar(Max) = 'USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
                    + Upper(@Company)
                    + ''' = ''ALL''
            BEGIN
                Insert #Table1StockCode
                    ( Company, StockCode, [StockDescription],[PartCategory], [Active-
POFlag])
                Select @DBCode
                ,StockCode
                ,[Description]
                ,[PartCategory]
                , [ActivePOFlag] = Max(Case When [PMH].[PurchaseOrder] Is Null Then
0
                                       Else 1
                                  End)
                From InvMaster
        Left Join [dbo].[PorMasterDetail] [PMD]
            On PMD.[MStockCode]=[StockCode]
               And [PMD].[MOrderQty] <> [PMD].[MReceivedQty]
        Left Join [dbo].[PorMasterHdr] [PMH]
            On [PMD].[PurchaseOrder] = [PMH].[PurchaseOrder]
               And [PMH].[CancelledFlag] <> ''Y''
```

Author: Johnson, Chris

```
                And [PMH].[DatePoCompleted] Is Null
Group By [StockCode]
       , [Description]
       , [PartCategory]
           End
    End';

--execute script against each db, populating the base tables

                Exec [Process].[ExecForEachDB_WithTableCheck] @cmd = @SQL ,
                    @SchemaTablesToCheck = @ListOfTables;

                Insert  [Lookups].[StockCode]
                        ( [Company]
                        , [StockCode]
                        , [LastUpdated]
                        , [StockDescription]
                        , [PartCategory]
                        , [ActivePOFlag]
                        )
                        Select  [Company]
                              , [StockCode]
                              , @LastUpdated
                              , [StockDescription]
                              , [PartCategory]
                              , [ActivePOFlag]
                        From    [#Table1StockCode];

                If @PrevCheck = 1
                    Begin
                        Declare @CurrentCount Int
                          , @PreviousCount Int;

                        Select  @CurrentCount = Count(*)
                        From    [Lookups].[StockCode]
                        Where   [LastUpdated] = @LastUpdated;

                        Select  @PreviousCount = Count(*)
                        From    [Lookups].[StockCode]
                        Where   [LastUpdated] <> @LastUpdated;

                        If @PreviousCount > @CurrentCount
                            Begin
                                Delete  [Lookups].[StockCode]
                                Where   [LastUpdated] = @LastUpdated;
                                Print 'UspUpdate_StockCode - Count has gone down
since last run, no update applied';
                                Print 'Current Count = '
                                    + Cast(@CurrentCount As Varchar(5))
                                    + ' Previous Count = '
                                    + Cast(@PreviousCount As Varchar(5));
                            End;
                        If @PreviousCount <= @CurrentCount
                            Begin
                                Delete  [Lookups].[StockCode]
                                Where   [LastUpdated] <> @LastUpdated;
                                Print 'UspUpdate_StockCode - Update applied
```

Author: Johnson, Chris

```
successfully';
                                End;
                        End;
                    If @PrevCheck = 0
                        Begin
                            Delete  [Lookups].[StockCode]
                            Where   [LastUpdated] <> @LastUpdated;
                            Print 'UspUpdate_StockCode - Update applied successfully';
                        End;
                End;
    End;
    If DateDiff(Minute , @LastDate , GetDate()) <= ( @HoursBetweenUpdates * 60 )
        Begin
            Print 'UspUpdate_StockCode - Table was last updated at '
                + Cast(@LastDate As Varchar(255)) + ' no update applied';
        End;


GO
EXEC sp_addextendedproperty N'MS_Description', N'Stored proc to update specified
table', 'SCHEMA', N'Process', 'PROCEDURE', N'UspUpdate_StockCode', NULL, NULL
GO
```

Uses

[Lookups].[StockCode]
[Process].[ExecForEachDB_WithTableCheck]
[Process]

Author: Johnson, Chris

[Process].[UspUpdate_TrnTypeModifier]

MS_Description

Stored proc to update specified table

Parameters

Name	Data Type	Max Length (Bytes)
@PrevCheck	int	4
@HoursBetweenUpdates	numeric(5,2)	5

SQL Script

```
CREATE Proc [Process].[UspUpdate_TrnTypeModifier]
    (
      @PrevCheck Int --if count is less than previous don't update

    , @HoursBetweenUpdates Numeric(5 , 2)
    )
As
    Begin
        Set NoCount On;

--check if table exists and create if it doesn't

        If ( Not Exists ( Select    1
                          From      [INFORMATION_SCHEMA].[TABLES]
                          Where     [TABLE_SCHEMA] = 'Lookups'
                                    And [TABLE_NAME] = 'TrnTypeAmountModifier' )
           )
            Begin
                Create Table [Lookups].[TrnTypeAmountModifier]
                    (
                      [Company] Varchar(150)    Collate Latin1_General_BIN
                    , [TrnType] Char(5)            Collate Latin1_General_BIN
                    , [AmountModifier] Int
                    , [LastUpdated] DateTime2
                    );
            End;


--check last time run and update if it's been longer than @HoursBetweenUpdates hours

        Declare @LastDate DateTime2;

        Select  @LastDate = Max([LastUpdated])
        From    [Lookups].[TrnTypeAmountModifier];

        If @LastDate Is Null
            Or DateDiff(Minute , @LastDate , GetDate()) > (@HoursBetweenUpdates*60)
            Begin
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Process.UspUpdate_Trn-TypeModifier

```
    --Set time of run

                Declare @LastUpdated DateTime2;
                Select  @LastUpdated = GetDate();

    --create master list of how codes affect stock

                Create Table [#AmountsTTM]
                    (
                      [TrnType] Char(5)        Collate Latin1_General_BIN
                    , [amountmodifier] Int
                    );

                Insert  [#AmountsTTM]
                        ( [TrnType]
                        , [amountmodifier]
                        )
                        Select  [t].[TrnType]
                              , [t].[AmountModifier]
                        From    ( Select    [TrnType] = 'R'
                                          , [AmountModifier] = 1
                                  Union
                                  Select    [TrnType] = 'I'
                                          , [AmountModifier] = -1
                                  Union
                                  Select    [TrnType] = 'P'
                                          , [AmountModifier] = 0
                                  Union
                                  Select    [TrnType] = 'T'
                                          , [AmountModifier] = 1
                                  Union
                                  Select    [TrnType] = 'A'
                                          , [AmountModifier] = 1
                                  Union
                                  Select    [TrnType] = 'C'
                                          , [AmountModifier] = 0
                                  Union
                                  Select    [TrnType] = 'M'
                                          , [AmountModifier] = 0
                                  Union
                                  Select    [TrnType] = 'B'
                                          , [AmountModifier] = 1
                                  Union
                                  Select    [TrnType] = 'S'
                                          , [AmountModifier] = -1
                                  Union
                                  Select    [TrnType] = 'D'
                                          , [AmountModifier] = -1
                                ) [t];

    --Get list of all companies in use


    --create temporary tables to be pulled from different databases, including a
column to id

                Create Table [#Table1TTM]
```

Author: Johnson, Chris

```
                    (
                      [CompanyName] Varchar(150)
                    );

    --create script to pull data from each db into the tables

                Declare @SQL Varchar(Max) = '
        USE [?];
        Declare @DB varchar(150),@DBCode varchar(150)
        Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end
        IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
        BEGIN
        Insert #Table1TTM
            ( CompanyName )
        Select @DBCode
        End';

    --execute script against each db, populating the base tables

                Exec [Process].[ExecForEachDB] @cmd = @SQL;

    --all companies process the same way

                Select  [T].[CompanyName]
                      , [A].[TrnType]
                      , [A].[amountmodifier]
                Into    [#ResultsTTm]
                From    [#Table1TTM] [T]
                        Left Join [#AmountsTTM] [A]
                            On 1 = 1;

    --placeholder for anomalous results that are different to master list


                Insert  [Lookups].[TrnTypeAmountModifier]
                        ( [Company]
                        , [TrnType]
                        , [AmountModifier]
                        , [LastUpdated]
                        )
                        Select  [CompanyName]
                              , [TrnType]
                              , [amountmodifier]
                              , @LastUpdated
                        From    [#ResultsTTm];

                If @PrevCheck = 1
                    Begin
                        Declare @CurrentCount Int
                          , @PreviousCount Int;

                        Select  @CurrentCount = Count(*)
                        From    [Lookups].[TrnTypeAmountModifier]
                        Where   [LastUpdated] = @LastUpdated;

                        Select  @PreviousCount = Count(*)
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Process.UspUpdate_Trn-TypeModifier

```
                        From    [Lookups].[TrnTypeAmountModifier]
                        Where   [LastUpdated] <> @LastUpdated;

                        If @PreviousCount > @CurrentCount
                            Begin
                                Delete  [Lookups].[TrnTypeAmountModifier]
                                Where   [LastUpdated] = @LastUpdated;
                                Print 'UspUpdate_TrnTypeModifier - Count has gone
down since last run, no update applied';
                                Print 'Current Count = '
                                    + Cast(@CurrentCount As Varchar(5))
                                    + ' Previous Count = '
                                    + Cast(@PreviousCount As Varchar(5));
                            End;
                        If @PreviousCount <= @CurrentCount
                            Begin
                                Delete  [Lookups].[TrnTypeAmountModifier]
                                Where   [LastUpdated] <> @LastUpdated;
                                Print 'UspUpdate_TrnTypeModifier - Update applied
successfully';
                            End;
                    End;
                If @PrevCheck = 0
                    Begin
                        Delete  [Lookups].[TrnTypeAmountModifier]
                        Where   [LastUpdated] <> @LastUpdated;
                        Print 'UspUpdate_TrnTypeModifier - Update applied
successfully';
                    End;
            End;
    End;
    If DateDiff(Minute , @LastDate , GetDate()) <= (@HoursBetweenUpdates*60)
        Begin
            Print 'UspUpdate_TrnTypeModifier - Table was last updated at '
                + Cast(@LastDate As Varchar(255)) + ' no update applied';
        End;
GO
EXEC sp_addextendedproperty N'MS_Description', N'Stored proc to update specified
table', 'SCHEMA', N'Process', 'PROCEDURE', N'UspUpdate_TrnTypeModifier', NULL, NULL
GO
```

Uses

[Lookups].[TrnTypeAmountModifier]
[Process].[ExecForEachDB]
[Process]

Author: Johnson, Chris

[Process].[UspUpdate_Warehouse]

MS_Description

Stored proc to update specified table

Parameters

Name	Data Type	Max Length (Bytes)
@PrevCheck	int	4
@HoursBetweenUpdates	numeric(5,2)	5

SQL Script

```
CREATE Proc [Process].[UspUpdate_Warehouse]
    (
      @PrevCheck Int
    , @HoursBetweenUpdates Numeric(5 , 2)
    )
As
    Begin

--remove nocount on to speed up query

        Set NoCount On;

--check if table exists and create if it doesn't

        If ( Not Exists ( Select    1
                          From      [INFORMATION_SCHEMA].[TABLES]
                          Where     [TABLE_SCHEMA] = 'Lookups'
                                    And [TABLE_NAME] = 'Warehouse' )
           )
            Begin
                Create Table [Lookups].[Warehouse]
                    (
                      [Company] Varchar(150)                collate Latin1_General_-
BIN
                    , [Warehouse] Varchar(150)                collate Latin1_-
General_BIN
                    , [WarehouseDescription] Varchar(200)     collate Latin1_General_-
BIN
                    , [LastUpdated] DateTime2
                    );
            End;



--check last time run and update if it's been longer than @HoursBetweenUpdates hours

        Declare @LastDate DateTime2;
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Process.UspUpdate_-Warehouse

```
        Select  @LastDate = Max([LastUpdated])
        From    [Lookups].[Warehouse];

        If @LastDate Is Null
            Or DateDiff(Minute , @LastDate , GetDate()) > ( @HoursBetweenUpdates
                                                            * 60 )
            Begin
    --Set time of run

                Declare @LastUpdated DateTime2;
                Select  @LastUpdated = GetDate();

--list the tables that are to be pulled back from each DB - if they are not found
the script will not be run against that db

                Declare @ListOfTables Varchar(Max) = 'InvMovements';

--create temporary tables to be pulled from different databases, including a column
to id

                Create Table [#Table1Wh]
                    (
                      [Company] Varchar(150)    collate Latin1_General_BIN
                    , [Warehouse] Varchar(150)    collate Latin1_General_BIN
                    );

--create script to pull data from each db into the tables

                Declare @Company Varchar(30) = 'All';
                Declare @SQL Varchar(Max) = 'USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
                    + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables
                    + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'',''))
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
                Insert #Table1Wh
                    ( Company, Warehouse)
                Select distinct @DBCode
                ,Warehouse
                From InvMovements
            End
    End';


--execute script against each db, populating the base tables
```

Author: Johnson, Chris

```
                Exec [Process].[ExecForEachDB] @cmd = @SQL;

                Insert  [Lookups].[Warehouse]
                        ( [Company]
                        , [Warehouse]
                        , [LastUpdated]
                        , [WarehouseDescription]
                        )
                        Select  [Company]
                              , [Warehouse]
                              , @LastUpdated
                              , Case When [Warehouse] = 'CE'
                                     Then 'Certified Environment'
                                     When [Warehouse] = 'RL' Then 'Red Label'
                                     When [Warehouse] = 'ZV' Then 'Zero Value'
                                     When [Warehouse] = 'RM'
                                     Then 'Raw Materials'
                                     When [Warehouse] = 'QU' Then 'Quarantine'
                                     When [Warehouse] = 'CB' Then 'Cambridge'
                                     When [Warehouse] = 'FG'
                                     Then 'Finished Goods'
                                End
                        From    [#Table1Wh];

                If @PrevCheck = 1
                    Begin
                        Declare @CurrentCount Int
                          , @PreviousCount Int;

                        Select  @CurrentCount = Count(*)
                        From    [Lookups].[Warehouse]
                        Where   [LastUpdated] = @LastUpdated;

                        Select  @PreviousCount = Count(*)
                        From    [Lookups].[Warehouse]
                        Where   [LastUpdated] <> @LastUpdated;

                        If @PreviousCount > @CurrentCount
                            Begin
                                Delete  [Lookups].[Warehouse]
                                Where   [LastUpdated] = @LastUpdated;
                                Print 'UspUpdate_Warehouse - Count has gone down
since last run, no update applied';
                                Print 'Current Count = '
                                    + Cast(@CurrentCount As Varchar(5))
                                    + ' Previous Count = '
                                    + Cast(@PreviousCount As Varchar(5));
                            End;
                        If @PreviousCount <= @CurrentCount
                            Begin
                                Delete  [Lookups].[Warehouse]
                                Where   [LastUpdated] <> @LastUpdated;
                                Print 'UspUpdate_Warehouse - Update applied
successfully';
                            End;
                    End;
```

Author: Johnson, Chris

```
                If @PrevCheck = 0
                    Begin
                        Delete  [Lookups].[Warehouse]
                        Where   [LastUpdated] <> @LastUpdated;
                        Print 'UspUpdate_Warehouse - Update applied successfully';
                    End;
            End;
    End;
    If DateDiff(Minute , @LastDate , GetDate()) <= ( @HoursBetweenUpdates * 60 )
        Begin
            Print 'UspUpdate_Warehouse - Table was last updated at '
                + Cast(@LastDate As Varchar(255)) + ' no update applied';
        End;


GO
EXEC sp_addextendedproperty N'MS_Description', N'Stored proc to update specified
table', 'SCHEMA', N'Process', 'PROCEDURE', N'UspUpdate_Warehouse', NULL, NULL
GO
```

Uses

[Lookups].[Warehouse]
[Process].[ExecForEachDB]
[Process]

Author: Johnson, Chris

[Report].[Results_AllCurrencyRates]

MS_Description

provides a list of currency rates for all time

Parameters

Name	Data Type	Max Length (Bytes)
@RedTagType	char	1
@RedTagUse	varchar(500)	500

SQL Script

```
Create Proc [Report].[Results_AllCurrencyRates]
    (
      @RedTagType Char(1)
    , @RedTagUse Varchar(500)
    )
As
    Begin
/*
Template designed by Chris Johnson, Prometic Group March 2016

query on currency (required to capture tags)
*/
--Red tag

        Declare @RedTagDB Varchar(255)= Db_Name();
        Exec [Process].[UspInsert_RedTagLogs] @StoredProcDb = 'BlackBox' ,
            @StoredProcSchema = 'Report' ,
            @StoredProcName = 'Results_AllCurrencyRates' ,
            @UsedByType = @RedTagType , @UsedByName = @RedTagUse ,
            @UsedByDb = @RedTagDB;

        Select  [CR].[StartDateTime]
              , [CR].[EndDateTime]
              , [CR].[Currency]
              , [CR].[CADDivision]
              , [CR].[CHFDivision]
              , [CR].[EURDivision]
              , [CR].[GBPDivision]
              , [CR].[JPYDivision]
              , [CR].[USDDivision]
              , [CR].[CADMultiply]
              , [CR].[CHFMultiply]
              , [CR].[EURMultiply]
              , [CR].[GBPMultiply]
              , [CR].[JPYMultiply]
              , [CR].[USDMultiply]
              , [CR].[LastUpdated]
        From    [Lookups].[CurrencyRates] As [CR];
```

Author: Johnson, Chris

```
    End;
GO
EXEC sp_addextendedproperty N'MS_Description', N'provides a list of currency rates
for all time', 'SCHEMA', N'Report', 'PROCEDURE', N'Results_AllCurrencyRates', NULL,
NULL
GO
```

Uses

[Lookups].[CurrencyRates]

[Process].[UspInsert_RedTagLogs]

[Report]

[Report].[Results_BudgetTest]

MS_Description

unused procedure

Parameters

Name	Data Type	Max Length (Bytes)
@TransJournal	varchar(150)	150

SQL Script

```
CREATE Proc [Report].[Results_BudgetTest]
    (
      @TransJournal VARCHAR(150)
    )
As
--exec Report.Results_BudgetTest @TransJournal  = 'Journal'

--exec Report.Results_BudgetTest @TransJournal  = 'Trans'

    Begin
--Declare @TransJournal VARCHAR(150) = 'Journal';


--Budget Data

        Select
            [bt].[BudgetTypeDesc]
          , [GB].[BudgetNumber]
          , [GB].[Company]
          , [GB].[DateLastModified]
          , [GB].[GlCode]
          , [GB].[Budget1]
          , [GB].[Budget2]
          , [GB].[Budget3]
          , [GB].[Budget4]
          , [GB].[Budget5]
          , [GB].[Budget6]
          , [GB].[Budget7]
          , [GB].[Budget8]
          , [GB].[Budget9]
          , [GB].[Budget10]
          , [GB].[Budget11]
          , [GB].[Budget12]
          , [GB].[Budget13]
          , [GB].[Budget14]
          , [GB].[Budget15]
        Into
            #Budgets
        From
```

Author: Johnson, Chris

```
            [SysproCompany40].dbo.GenBudgets GB
        Left Join [BlackBox].[Lookups].[BudgetType] As [bt]
            On [bt].[BudgetType] = [GB].[BudgetType]
        Where
            [Company] = '10'
        Order By
            [bt].[BudgetTypeDesc];

--Pick up either Transactions from 40 or journal details

        If @TransJournal = 'Trans'
            Begin
    --Transactions

                Select
                    [gt].[GlYear]
                  , [gts].[SourceDesc]
                  , [GlCode]
                  , [gt].[Journal]
                  , [gt].[Reference]
                  , [gt].[Comment]
                  , [gt].[JnlDate]
                  , [EntryPosted] = Null
                  , [Period1] = [gt].[1]
                  , [Period2] = [gt].[2]
                  , [Period3] = [gt].[3]
                  , [Period4] = [gt].[4]
                  , [Period5] = [gt].[5]
                  , [Period6] = [gt].[6]
                  , [Period7] = [gt].[7]
                  , [Period8] = [gt].[8]
                  , [Period9] = [gt].[9]
                  , [Period10] = [gt].[10]
                  , [Period11] = [gt].[11]
                  , [Period12] = [gt].[12]
                  , [Period13] = [gt].[13]
                  , [Period14] = [gt].[14]
                  , [Period15] = [gt].[15]
                Into
                    #Transpivoted
                From
                    [SysproCompany40].[dbo].[GenTransaction] Pivot ( SUM([Entry-
Value]) For [GlPeriod] In ( [1],
                                                                    [2], [3], [4],
                                                                    [5], [6], [7],
                                                                    [8], [9], [10],
                                                                    [11], [12], [13],
                                                                    [14], [15] ) ) As [gt]
                Left Join [BlackBox].[Lookups].[GenTransactionSource] As [gts]
                    On [gts].[Source] = [gt].[Source]
                Where
                    [Journal] = 2726;

                Select
                    [BudgetTypeDesc] = COALESCE([BudgetTypeDesc],
                                                'No Budget')
                  , [BudgetNumber]
```

Author: Johnson, Chris

```
                    , [b].[Company]
                    , [b].[DateLastModified]
                    , [GlCode] = COALESCE([b].[GlCode], [t].[GlCode])
                    , [gm].[Description]
                    , [t].[GlYear]
                    , [t].[SourceDesc]
                    , [t].[Journal]
                    , [Reference] = Case When [t].[Reference] = '' Then Null
                                        Else [t].[Reference]
                                    End
                    , [Comment] = Case When [t].[Comment] = '' Then Null
                                       Else [t].[Comment]
                                  End
                    , [EntryPosted]= null
                    , [t].[JnlDate]
                    , [Budget1]  =coalesce([Budget1] ,0)
                    , [Period1]  =coalesce([Period1] ,0)
                    , [Budget2]  =coalesce([Budget2] ,0)
                    , [Period2]  =coalesce([Period2] ,0)
                    , [Budget3]  =coalesce([Budget3] ,0)
                    , [Period3]  =coalesce([Period3] ,0)
                    , [Budget4]  =coalesce([Budget4] ,0)
                    , [Period4]  =coalesce([Period4] ,0)
                    , [Budget5]  =coalesce([Budget5] ,0)
                    , [Period5]  =coalesce([Period5] ,0)
                    , [Budget6]  =coalesce([Budget6] ,0)
                    , [Period6]  =coalesce([Period6] ,0)
                    , [Budget7]  =coalesce([Budget7] ,0)
                    , [Period7]  =coalesce([Period7] ,0)
                    , [Budget8]  =coalesce([Budget8] ,0)
                    , [Period8]  =coalesce([Period8] ,0)
                    , [Budget9]  =coalesce([Budget9] ,0)
                    , [Period9]  =coalesce([Period9] ,0)
                    , [Budget10] =coalesce([Budget10],0)
                    , [Period10] =coalesce([Period10],0)
                    , [Budget11] =coalesce([Budget11],0)
                    , [Period11] =coalesce([Period11],0)
                    , [Budget12] =coalesce([Budget12],0)
                    , [Period12] =coalesce([Period12],0)
                    , [Budget13] =coalesce([Budget13],0)
                    , [Period13] =coalesce([Period13],0)
                    , [Budget14] =coalesce([Budget14],0)
                    , [Period14] =coalesce([Period14],0)
                    , [Budget15] =coalesce([Budget15],0)
                    , [Period15] =coalesce([Period15],0)
                From
                    [#Budgets] As [b]
                Full Outer Join [#Transpivoted] As [t]
                    On [b].[GlCode] = [t].[GlCode]
                Left Join [SysproCompany40].[dbo].[GenMaster] As [gm]
                    On [gm].[GlCode] = COALESCE([b].[GlCode], [t].[GlCode])
                Union
                Select
                    [BudgetTypeDesc]
                    , [BudgetNumber]
                    , [b].[Company]
                    , [b].[DateLastModified]
```

Author: Johnson, Chris

```
                  , [GlCode] = [b].[GlCode]
                  , [gm].[Description]
                  , [GlYear] = Null
                  , [SourceDesc] = Null
                  , [Journal] = Null
                  , [Reference] = Null
                  , [Comment] = Null
                  , [EntryPosted]= null
                  , [JnlDate] = Null
                  , [Budget1]  =coalesce([Budget1] ,0)
                  , [Period1]  =null
                  , [Budget2]  =coalesce([Budget2] ,0)
                  , [Period2]  =null
                  , [Budget3]  =coalesce([Budget3] ,0)
                  , [Period3]  =null
                  , [Budget4]  =coalesce([Budget4] ,0)
                  , [Period4]  =null
                  , [Budget5]  =coalesce([Budget5] ,0)
                  , [Period5]  =null
                  , [Budget6]  =coalesce([Budget6] ,0)
                  , [Period6]  =null
                  , [Budget7]  =coalesce([Budget7] ,0)
                  , [Period7]  =null
                  , [Budget8]  =coalesce([Budget8] ,0)
                  , [Period8]  =null
                  , [Budget9]  =coalesce([Budget9] ,0)
                  , [Period9]  =null
                  , [Budget10] =coalesce([Budget10],0)
                  , [Period10] =null
                  , [Budget11] =coalesce([Budget11],0)
                  , [Period11] =null
                  , [Budget12] =coalesce([Budget12],0)
                  , [Period12] =null
                  , [Budget13] =coalesce([Budget13],0)
                  , [Period13] =null
                  , [Budget14] =coalesce([Budget14],0)
                  , [Period14] =null
                  , [Budget15] =coalesce([Budget15],0)
                  , [Period15] =null
              From
                  [#Budgets] As [b]
              Left Join [SysproCompany40].[dbo].[GenMaster] As [gm]
                  On [gm].[GlCode] = [b].[GlCode];

              Drop Table [#Budgets];
              Drop Table [#Transpivoted];
          End;
      If @TransJournal = 'Journal'
          Begin
              Select
                  [EntryValue] = [gjd].[EntryValue]
                  * Case When gjd.[EntryType] = 'D' Then 1
                         When gjd.[EntryType] = 'C' Then -1
                         Else 0
                    End
                , [gjd].[GlCode]
                , [gjd].[Journal]
```

Author: Johnson, Chris

```
                  , [gjd].[GlPeriod]
                  , [gjd].[GlYear]
                  , [gjd].[EntryPosted]
                  , [gts].[SourceDesc]
                  , JnlDate = [gjd].[EntryDate]
                  , [Reference] = Case When [gjd].[Reference] = '' Then Null
                                       Else [gjd].[Reference]
                                  End
                  , [Comment] = Case When [gjd].[Comment] = '' Then Null
                                     Else [gjd].[Comment]
                                End
              Into
                  #JD_raw
              From
                  [SysproCompany10]..[GenJournalDetail] gjd
              Left Join [BlackBox].[Lookups].[GenTransactionSource] As [gts]
                  On [gts].[Source] = [gjd].[Source];
              Select
                  GlCode
                , [GlYear]
                , [JnlDate]
                , [Journal]
                , [SourceDesc]
                , Reference
                , Comment
                , [EntryPosted]
                , Period1 = [1]
                , Period2 = [2]
                , Period3 = [3]
                , Period4 = [4]
                , Period5 = [5]
                , Period6 = [6]
                , Period7 = [7]
                , Period8 = [8]
                , Period9 = [9]
                , Period10 = [10]
                , Period11 = [11]
                , Period12 = [12]
                , Period13 = [13]
                , Period14 = [14]
                , Period15 = [15]
              Into
                  #JournalDetail
              From
                  [#JD_raw] Pivot ( SUM([EntryValue]) For [GlPeriod] In ( [1],
                                                                          [2], [3], [4],
                                                                          [5], [6], [7],
                                                                          [8], [9], [10],
                                                                          [11], [12], [13],
                                                                          [14], [15] ) ) As
[jr];
              Select
                  [BudgetTypeDesc] = COALESCE([BudgetTypeDesc],
                                              'No Budget')
                , [BudgetNumber]
                , [b].[Company]
                , [b].[DateLastModified]
```

Author: Johnson, Chris

```
                    , [GlCode] = COALESCE([b].[GlCode], [t].[GlCode])
                    , [gm].[Description]
                    , [t].[GlYear]
                    , [t].[SourceDesc]
                    , [t].[Journal]
                    , [Reference] = Case When [t].[Reference] = '' Then Null
                                        Else [t].[Reference]
                                    End
                    , [Comment] = Case When [t].[Comment] = '' Then Null
                                       Else [t].[Comment]
                                  End
                    , [t].[EntryPosted]
                    , [t].[JnlDate]
                    , [Budget1]  = coalesce([Budget1]    ,0)
                    , [Period1]  = coalesce([Period1]    ,0)
                    , [Budget2]  = coalesce([Budget2]    ,0)
                    , [Period2]  = coalesce([Period2]    ,0)
                    , [Budget3]  = coalesce([Budget3]    ,0)
                    , [Period3]  = coalesce([Period3]    ,0)
                    , [Budget4]  = coalesce([Budget4]    ,0)
                    , [Period4]  = coalesce([Period4]    ,0)
                    , [Budget5]  = coalesce([Budget5]    ,0)
                    , [Period5]  = coalesce([Period5]    ,0)
                    , [Budget6]  = coalesce([Budget6]    ,0)
                    , [Period6]  = coalesce([Period6]    ,0)
                    , [Budget7]  = coalesce([Budget7]    ,0)
                    , [Period7]  = coalesce([Period7]    ,0)
                    , [Budget8]  = coalesce([Budget8]    ,0)
                    , [Period8]  = coalesce([Period8]    ,0)
                    , [Budget9]  = coalesce([Budget9]    ,0)
                    , [Period9]  = coalesce([Period9]    ,0)
                    , [Budget10] = coalesce([Budget10],0)
                    , [Period10] = coalesce([Period10],0)
                    , [Budget11] = coalesce([Budget11],0)
                    , [Period11] = coalesce([Period11],0)
                    , [Budget12] = coalesce([Budget12],0)
                    , [Period12] = coalesce([Period12],0)
                    , [Budget13] = coalesce([Budget13],0)
                    , [Period13] = coalesce([Period13],0)
                    , [Budget14] = coalesce([Budget14],0)
                    , [Period14] = coalesce([Period14],0)
                    , [Budget15] = coalesce([Budget15],0)
                    , [Period15] = coalesce([Period15],0)
                From
                    [#Budgets] As [b]
                Full Outer Join [#JournalDetail] As [t]
                    On [b].[GlCode] = t.[GlCode]
                Left Join [SysproCompany40].[dbo].[GenMaster] As [gm]
                    On [gm].[GlCode] = COALESCE([b].[GlCode], [t].[GlCode])
                Union
                Select
                    [BudgetTypeDesc]
                  , [BudgetNumber]
                  , [b].[Company]
                  , [b].[DateLastModified]
                  , [GlCode] = [b].[GlCode]
                  , [gm].[Description]
```

Author: Johnson, Chris

```
                    , [GlYear] = Null
                    , [SourceDesc] = Null
                    , [Journal] = Null
                    , [Reference] = Null
                    , [Comment] = Null
                    , [EntryPosted]= null
                    , [JnlDate] = Null
                    , [Budget1]  =coalesce([Budget1] ,0)
                    , [Period1]  =null
                    , [Budget2]  =coalesce([Budget2] ,0)
                    , [Period2]  =null
                    , [Budget3]  =coalesce([Budget3] ,0)
                    , [Period3]  =null
                    , [Budget4]  =coalesce([Budget4] ,0)
                    , [Period4]  =null
                    , [Budget5]  =coalesce([Budget5] ,0)
                    , [Period5]  =null
                    , [Budget6]  =coalesce([Budget6] ,0)
                    , [Period6]  =null
                    , [Budget7]  =coalesce([Budget7] ,0)
                    , [Period7]  =null
                    , [Budget8]  =coalesce([Budget8] ,0)
                    , [Period8]  =null
                    , [Budget9]  =coalesce([Budget9] ,0)
                    , [Period9]  =null
                    , [Budget10] =coalesce([Budget10],0)
                    , [Period10] =null
                    , [Budget11] =coalesce([Budget11],0)
                    , [Period11] =null
                    , [Budget12] =coalesce([Budget12],0)
                    , [Period12] =null
                    , [Budget13] =coalesce([Budget13],0)
                    , [Period13] =null
                    , [Budget14] =coalesce([Budget14],0)
                    , [Period14] =null
                    , [Budget15] =coalesce([Budget15],0)
                    , [Period15] =null
                From
                    [#Budgets] As [b]
                Left Join [SysproCompany40].[dbo].[GenMaster] As [gm]
                    On [gm].[GlCode] = [b].[GlCode];

                Drop Table [#Budgets];
                Drop Table [#JD_raw];
                Drop Table [#JournalDetail];
            End;
    End;
GO
EXEC sp_addextendedproperty N'MS_Description', N'unused procedure', 'SCHEMA',
N'Report', 'PROCEDURE', N'Results_BudgetTest', NULL, NULL
GO
```

Uses

[Lookups].[BudgetType]

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.Results_Budget-Test

[Lookups].[GenTransactionSource]

[Report]

Author: Johnson, Chris

[Report].[UspResults_AccPayable_AgedInvoices]

MS_Description

returns details of aged accounts payable invoices

Parameters

Name	Data Type	Max Length (Bytes)
@Company	varchar(max)	max
@RedTagType	char	1
@RedTagUse	varchar(500)	500

SQL Script

```
CREATE Proc [Report].[UspResults_AccPayable_AgedInvoices]
    (
      @Company Varchar(Max)
    , @RedTagType Char(1)
    , @RedTagUse Varchar(500)
    )
As --Exec [Report].[UspResults_AccPayable_AgedInvoices]  10

    Begin
/*
Template designed by Chris Johnson, Prometic Group September 2015
Stored procedure set out to query multiple databases with the same information and
return it in a collated format
Replacement of aged AP analysis
*/
        Set NoCount Off;
        If IsNumeric(@Company) = 0
            Begin
                Select  @Company = Upper(@Company);
            End;

--remove nocount on to speed up query

        Set NoCount On;
--Red tag

        Declare @RedTagDB Varchar(255)= Db_Name();
        Exec [Process].[UspInsert_RedTagLogs] @StoredProcDb = 'BlackBox' ,
            @StoredProcSchema = 'Report' ,
            @StoredProcName = 'UspResults_AccPayable_AgedInvoices' ,
            @UsedByType = @RedTagType , @UsedByName = @RedTagUse ,
            @UsedByDb = @RedTagDB;
--list the tables that are to be pulled back from each DB - if they are not found
the script will not be run against that db

        Declare @ListOfTables Varchar(Max) = 'ApSupplier,ApInvoice';
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_Acc-Payable_AgedInvoices

```
--create temporary tables to be pulled from different databases, including a column
to id

        Create Table [#ApSupplier]
            (
              [DatabaseName] Varchar(150)        collate latin1_general_bin
            , [Supplier] Varchar(50)            collate latin1_general_bin
            , [SupplierName] Varchar(200)        collate latin1_general_bin
            );
        Create Table [#ApInvoice]
            (
              [DatabaseName] Varchar(150)        collate latin1_general_bin
            , [Invoice] Varchar(50)               collate latin1_general_bin
            , [InvoiceDate] DateTime2
            , [OrigInvValue] Numeric(20 , 7)
            , [OrigDiscValue] Numeric(20 , 7)
            , [MthInvBal1] Numeric(20 , 7)
            , [MthInvBal2] Numeric(20 , 7)
            , [MthInvBal3] Numeric(20 , 7)
            , [Currency] Char(3)                collate latin1_general_bin
            , [ConvRate] Numeric(20 , 7)
            , [PostCurrency] Char(3)            collate latin1_general_bin
            , [Supplier] Varchar(50)            collate latin1_general_bin
            );

--create script to pull data from each db into the tables

        Declare @SQL1 Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end'
            + --Only query DBs beginning SysProCompany

            '
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN'
            + --only companies selected in main run, or if companies selected then
all

            '
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT'
            + --count number of tables requested (number of commas plus one)

            '
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')'
            + --Count of the tables requested how many exist in the db

            '
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'','')) '
            + --only if the count matches (all the tables exist in the requested db)
```

Author: Johnson, Chris

```
then run the script

            '
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
                Insert [#ApInvoice]
                        ( [DatabaseName]
                        , [Invoice]
                        , [InvoiceDate]
                        , [OrigInvValue]
                        , [OrigDiscValue]
                        , [MthInvBal1]
                        , [MthInvBal2]
                        , [MthInvBal3]
                        , [Currency]
                        , [ConvRate]
                        , [PostCurrency]
                        , [Supplier]
                        )
                SELECT [DatabaseName]=@DBCode
                     , [ai].[Invoice]
                     , [ai].[InvoiceDate]
                     , [ai].[OrigInvValue]
                     , [ai].[OrigDiscValue]
                     , [ai].[MthInvBal1]
                     , [ai].[MthInvBal2]
                     , [ai].[MthInvBal3]
                     , [ai].[Currency]
                     , [ai].[ConvRate]
                     , [ai].[PostCurrency]
                     , [ai].[Supplier]
                From [ApInvoice] As [ai]
            End
    End';
        Declare @SQL2 Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end'
            + --Only query DBs beginning SysProCompany

            '
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN'
            + --only companies selected in main run, or if companies selected then
all

            '
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT'
            + --count number of tables requested (number of commas plus one)

            '
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')'
```

Author: Johnson, Chris

```
            + --Count of the tables requested how many exist in the db

            '
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'',')) '
            + --only if the count matches (all the tables exist in the requested db)
then run the script

            '
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
                Insert [#ApSupplier]
                    ( [DatabaseName]
                    , [Supplier]
                    , [SupplierName]
                    )
                SELECT [DatabaseName]=@DBCode
                     , [as].[Supplier]
                     , [as].[SupplierName]
                From [ApSupplier] As [as]
            End
    End';
--Enable this function to check script changes (try to run script directly against
db manually)

--Print @SQL


--execute script against each db, populating the base tables

        Exec [sys].[sp_MSforeachdb] @SQL1;
        Exec [sys].[sp_MSforeachdb] @SQL2;

--define the results you want to return

        Create Table [#Results]
            (
              [DatabaseName] Varchar(150)        collate latin1_general_bin
            , [Supplier] Varchar(50)            collate latin1_general_bin
            , [SupplierName] Varchar(150)        collate latin1_general_bin
            , [Invoice] Varchar(50)               collate latin1_general_bin
            , [InvoiceDate] DateTime2
            , [OrigInvValue] Numeric(20 , 7)
            , [OrigDiscValue] Numeric(20 , 7)
            , [MthInvBal1] Numeric(20 , 7)
            , [MthInvBal2] Numeric(20 , 7)
            , [MthInvBal3] Numeric(20 , 7)
            , [Currency] Char(3)                collate latin1_general_bin
            , [ConvRate] Numeric(20 , 7)
            , [PostCurrency] Char(3)            collate latin1_general_bin
            );

--Placeholder to create indexes as required

--create NonClustered Index Index_Name On #Table1 (DatabaseName) Include (Column-
Name)
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_Acc-Payable_AgedInvoices

```
--script to combine base data and insert into results table

        Insert  [#Results]
                ( [DatabaseName]
                , [Supplier]
                , [SupplierName]
                , [Invoice]
                , [InvoiceDate]
                , [OrigInvValue]
                , [OrigDiscValue]
                , [MthInvBal1]
                , [MthInvBal2]
                , [MthInvBal3]
                , [Currency]
                , [ConvRate]
                , [PostCurrency]
                )
                Select  [asp].[DatabaseName]
                      , [asp].[Supplier]
                      , [asp].[SupplierName]
                      , [ai].[Invoice]
                      , [ai].[InvoiceDate]
                      , [ai].[OrigInvValue]
                      , [ai].[OrigDiscValue]
                      , [ai].[MthInvBal1]
                      , [ai].[MthInvBal2]
                      , [ai].[MthInvBal3]
                      , [ai].[Currency]
                      , [ai].[ConvRate]
                      , [ai].[PostCurrency]
                From    [#ApSupplier] As [asp]
                        Inner Join [#ApInvoice] As [ai]
                            On [ai].[Supplier] = [asp].[Supplier]
                               And [ai].[DatabaseName] = [asp].[DatabaseName];

--return results

        Select  [cn].[CompanyName]
              , [r].[Supplier]
              , [r].[SupplierName]
              , [r].[Invoice]
              , [InvoiceDate] = Cast([r].[InvoiceDate] As Date)
              , [r].[OrigInvValue]
              , [r].[OrigDiscValue]
              , [r].[MthInvBal1]
              , [r].[MthInvBal2]
              , [r].[MthInvBal3]
              , [r].[Currency]
              , [r].[ConvRate]
              , [r].[PostCurrency]
        From    [#Results] As [r]
                Left Join [Lookups].[CompanyNames] As [cn]
                    On [r].[DatabaseName] = [cn].[Company];

    End;
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_Acc-Payable_AgedInvoices

```
GO
EXEC sp_addextendedproperty N'MS_Description', N'returns details of aged accounts payable invoices', 'SCHEMA', N'Report', 'PROCEDURE', N'UspResults_AccPayable_Aged-Invoices', NULL, NULL
GO
```

Uses

[Lookups].[CompanyNames]

[Process].[UspInsert_RedTagLogs]

[Report]

Author: Johnson, Chris

[Report].[UspResults_AllBankBalances]

MS_Description

returns list of all bank balances

SQL Script

```
CREATE Proc [Report].[UspResults_AllBankBalances]
As
    Begin
/*
Template designed by Chris Johnson, Prometic Group September 2015
Stored procedure set out to query multiple databases with the same information and
return it in a collated format
--Exec [Report].[UspResults_AllBankBalances]
*/
        Declare @Company Varchar(Max)= 'All';

        Set NoCount On;

--list the tables that are to be pulled back from each DB - if they are not found
the script will not be run against that db

        Declare @ListOfTables Varchar(Max) = 'AssetDepreciation,TblApTerms';

--create temporary tables to be pulled from different databases, including a column
to id

        Create Table [#Base]
            (
              [DatabaseName] Varchar(150)    Collate Latin1_General_BIN
            , [BaseCcy] Char(3)                Collate Latin1_General_BIN
            , [BaseCcyName] Varchar(150)    Collate Latin1_General_BIN
            );
        Create Table [#SecondConvert]
            (
              [DatabaseName] Varchar(150)    Collate Latin1_General_BIN
            , [SecondConvert] Char(3)        Collate Latin1_General_BIN
            , [SecondConvertSellRate] Numeric(20 , 7)
            , [SecondConvertBuyRate] Numeric(20 , 7)
            );
        Create Table [#TblCurrency]
            (
              [DatabaseName] Varchar(150)    Collate Latin1_General_BIN
            , [Currency] Char(3)            Collate Latin1_General_BIN
            , [Description] Varchar(150)    Collate Latin1_General_BIN
            , [BuyExchangeRate] Numeric(20 , 7)
            , [SellExchangeRate] Numeric(20 , 7)
            );
        Create Table [#ApBank]
            (
              [DatabaseName] Varchar(150)    Collate Latin1_General_BIN
            , [Bank] Varchar(15)            Collate Latin1_General_BIN
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_All-BankBalances

```
            , [Description] Varchar(50)        Collate Latin1_General_BIN
            , [CashGlCode] Varchar(50)        collate latin1_general_bin
            , [Currency] Char(3)            collate latin1_general_bin
            , [CurrentBalance] Numeric(20 , 7)
            , [StatementBalance] Numeric(20 , 7)
            , [OutStandingDeposits] Numeric(20 , 7)
            , [OutStandingWithdrawals] Numeric(20 , 7)
            , [PrevMonth1CurrentBalance] Numeric(20 , 7)
            , [PrevMonth1StatementBalance] Numeric(20 , 7)
            , [PrevMonth1OutStandingDeposits] Numeric(20 , 7)
            , [PrevMonth1OutStandingWithdrawals] Numeric(20 , 7)
            , [PrevMonth2CurrentBalance] Numeric(20 , 7)
            , [PrevMonth2StatementBalance] Numeric(20 , 7)
            , [PrevMonth2OutStandingDeposits] Numeric(20 , 7)
            , [PrevMonth2OutStandingWithdrawals] Numeric(20 , 7)
            );

--create script to pull data from each db into the tables

        Declare @SQLBase Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end'
            + --Only query DBs beginning SysProCompany

            '
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN'
            + --only companies selected in main run, or if companies selected then
all

            '
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT'
            + --count number of tables requested (number of commas plus one)

            '
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')'
            + --Count of the tables requested how many exist in the db

            '
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'','')) '
            + --only if the count matches (all the tables exist in the requested db)
then run the script

            '
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
                Insert  [#Base]
                        ( [DatabaseName]
                        , [BaseCcy]
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_All-BankBalances

```
                        , [BaseCcyName]
                        )
                Select
                    @DBCode
                    , BaseCcy = [tc].[Currency]
                    , BaseCcyName = [tc].[Description]
                From
                    [dbo].[TblCurrency] As [tc]
                Where [tc].[BuyExchangeRate] = 1;
            End
    End';
        Declare @SQLSecondConvert Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end'
            + --Only query DBs beginning SysProCompany

            '
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN'
            + --only companies selected in main run, or if companies selected then
all

            '
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT'
            + --count number of tables requested (number of commas plus one)

            '
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')'
            + --Count of the tables requested how many exist in the db

            '
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'','')) '
            + --only if the count matches (all the tables exist in the requested db)
then run the script

            '
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
                Insert  [#SecondConvert]
                        ( [DatabaseName]
                        , [SecondConvert]
                        , [SecondConvertSellRate]
                        , [SecondConvertBuyRate]
                        )
                Select @DBCode
                    , SecondConvert = [tc].[Currency]
                    , SecondConvertSellRate = [tc].[SellExchangeRate]
                    , SecondConvertBuyRate = [tc].[BuyExchangeRate]
                From [dbo].[TblCurrency] As [tc];
```

Author: Johnson, Chris

```
            End
    End';
        Declare @SQLTblCurrency Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end'
            + --Only query DBs beginning SysProCompany

            '
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN'
            + --only companies selected in main run, or if companies selected then
all

            '
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT'
            + --count number of tables requested (number of commas plus one)

            '
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')'
            + --Count of the tables requested how many exist in the db

            '
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'','')) '
            + --only if the count matches (all the tables exist in the requested db)
then run the script

            '
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
                Insert  [#TblCurrency]
                ( [DatabaseName]
                , [Currency]
                , [Description]
                , [BuyExchangeRate]
                , [SellExchangeRate]
                )
                Select @DBCode
                  , [tc].[Currency]
                  , [tc].[Description]
                  , [tc].[BuyExchangeRate]
                  , [tc].[SellExchangeRate]
                From [TblCurrency] As [tc];
            End
    End';
        Declare @SQLApBank Varchar(Max) = 'USE [?];
Declare @DB varchar(150),@DBCode varchar(150)
Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end'
            + --Only query DBs beginning SysProCompany
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_All-BankBalances

```
            '
IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
BEGIN'
            + --only companies selected in main run, or if companies selected then
all

            '
IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables + '''
, @RequiredCountOfTables INT
, @ActualCountOfTables INT'
            + --count number of tables requested (number of commas plus one)

            '
Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_SplitString](@ListOf-
Tables,'','')'
            + --Count of the tables requested how many exist in the db

            '
Select @ActualCountOfTables = COUNT(1) FROM sys.tables
Where name In (Select Value Collate Latin1_General_BIN From BlackBox.dbo.udf_Split-
String(@ListOfTables,'',''))  '
            + --only if the count matches (all the tables exist in the requested db)
then run the script

            '
If @ActualCountOfTables=@RequiredCountOfTables
BEGIN
Insert  [#ApBank]
( [DatabaseName], [Bank]
, [Description], [CashGlCode]
, [Currency], [CurrentBalance]
, [StatementBalance], [OutStandingDeposits]
, [OutStandingWithdrawals], [PrevMonth1CurrentBalance]
, [PrevMonth1StatementBalance], [PrevMonth1OutStandingDeposits]
, [PrevMonth1OutStandingWithdrawals], [PrevMonth2CurrentBalance]
, [PrevMonth2StatementBalance], [PrevMonth2OutStandingDeposits]
, [PrevMonth2OutStandingWithdrawals]
)
Select
@DBCode, [Bank]
, [Description], [CashGlCode]
, [Currency], [CurrentBalance] = [CbCurBalLoc1]
, [StatementBalance] = [CbStmtBal1], [OutStandingDeposits] = OutstDep1
, [OutStandingWithdrawals] = OutstWith1, [PrevMonth1CurrentBalance] = CbCurBal2
, [PrevMonth1StatementBalance] = CbStmtBal2, [PrevMonth1OutStandingDeposits] = Outst-
Dep2
, [PrevMonth1OutStandingWithdrawals] = OutstWith2, [PrevMonth2CurrentBalance] = Cb-
CurBal3
, [PrevMonth2StatementBalance] = CbStmtBal3, [PrevMonth2OutStandingDeposits] = Outst-
Dep3
, [PrevMonth2OutStandingWithdrawals] = OutstWith3
From dbo.ApBank;
End
End';
--Enable this function to check script changes (try to run script directly against
db manually)
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_All-BankBalances

```
--Print @SQL


--execute script against each db, populating the base tables

--Print 1

        Exec [Process].[ExecForEachDB] @cmd = @SQLBase;
--Print 2

        Exec [Process].[ExecForEachDB] @cmd = @SQLSecondConvert;
--Print 3

        Exec [Process].[ExecForEachDB] @cmd = @SQLTblCurrency;
--Print 4

        Exec [Process].[ExecForEachDB] @cmd = @SQLApBank;
--Print 5


--define the results you want to return

        Create Table [#CurrencyConversions]
            (
              [DatabaseName] Varchar(150)            collate latin1_general_bin
            , [Currency] Char(3)                    collate latin1_general_bin
            , [Description] Varchar(150)            collate latin1_general_bin
            , [BuyExchangeRate] Numeric(20 , 7)
            , [SellExchangeRate] Numeric(20 , 7)
            , [BaseCcy] Char(3)                        collate latin1_general_bin
            , [BaseCcyName] Varchar(150)            Collate Latin1_General_BIN
            , [SecondConvertBuyRate] Numeric(20 , 7)
            , [SecondConvertSellRate] Numeric(20 , 7)
            , [SecondConvert] Char(3)                collate latin1_general_bin
            , [RateSell] Numeric(20 , 7)
            , [RateBuy] Numeric(20 , 7)
            );
        Create Table [#Results]
            (
              [DatabaseName] Varchar(150)            collate latin1_general_bin
            , [CompanyName] Varchar(150)            collate latin1_general_bin
            , [Bank] Varchar(10)                    collate latin1_general_bin
            , [BankDescription] Varchar(150)        collate latin1_general_bin
            , [CashGlCode] Varchar(150)                collate latin1_general_bin
            , [BankCurrency] Char(3)                collate latin1_general_bin
            , [CurrentBalance] Numeric(20 , 7)
            , [StatementBalance] Numeric(20 , 7)
            , [OutStandingDeposits] Numeric(20 , 7)
            , [OutStandingWithdrawals] Numeric(20 , 7)
            , [PrevMonth1CurrentBalance] Numeric(20 , 7)
            , [PrevMonth1StatementBalance] Numeric(20 , 7)
            , [PrevMonth1OutStandingDeposits] Numeric(20 , 7)
            , [PrevMonth1OutStandingWithdrawals] Numeric(20 , 7)
            , [PrevMonth2CurrentBalance] Numeric(20 , 7)
            , [PrevMonth2StatementBalance] Numeric(20 , 7)
            , [PrevMonth2OutStandingDeposits] Numeric(20 , 7)
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_All-BankBalances

```
            , [PrevMonth2OutStandingWithdrawals] Numeric(20 , 7)
            , [CurrencyDescription] Varchar(150)    collate latin1_general_bin
            , [BuyExchangeRate] Numeric(20 , 7)
            , [SellExchangeRate] Numeric(20 , 7)
            , [BaseCcy] Char(3)                       collate latin1_general_bin
            , [BaseCcyName] Varchar(150)            collate latin1_general_bin
            , [SecondConvertBuyRate] Numeric(20 , 7)
            , [SecondConvertSellRate] Numeric(20 , 7)
            , [SecondConvert] Char(3)                collate latin1_general_bin
            , [RateSell] Numeric(20 , 7)
            , [RateBuy] Numeric(20 , 7)
            );
--Placeholder to create indexes as required

--create NonClustered Index Index_Name On #Table1 (DatabaseName) Include (Column-Name)


--script to combine base data and insert into results table

        Print 6;
        Insert  [#CurrencyConversions]
                ( [DatabaseName]
                , [Currency]
                , [Description]
                , [BuyExchangeRate]
                , [SellExchangeRate]
                , [BaseCcy]
                , [BaseCcyName]
                , [SecondConvertBuyRate]
                , [SecondConvertSellRate]
                , [SecondConvert]
                , [RateSell]
                , [RateBuy]
                )
                Select  [tc].[DatabaseName]
                      , [tc].[Currency]
                      , [tc].[Description]
                      , [tc].[BuyExchangeRate]
                      , [tc].[SellExchangeRate]
                      , [b].[BaseCcy]
                      , [b].[BaseCcyName]
                      , [c].[SecondConvertBuyRate]
                      , [c].[SecondConvertSellRate]
                      , [c].[SecondConvert]
                      , [RateSell] = [tc].[SellExchangeRate]
                        / [c].[SecondConvertSellRate]
                      , [RateBuy] = [tc].[BuyExchangeRate]
                        / [c].[SecondConvertBuyRate]
                From    [#TblCurrency] As [tc]
                        Left Join [#Base] As [b] On [b].[DatabaseName] =
[tc].[DatabaseName]
                        Left Join [#SecondConvert] As [c] On [c].[DatabaseName] =
[tc].[DatabaseName]
                Order By [c].[SecondConvert];

        Print 7;
        Insert  [#Results]
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_All-BankBalances

```
                ( [DatabaseName]
                , [CompanyName]
                , [Bank]
                , [BankDescription]
                , [CashGlCode]
                , [BankCurrency]
                , [CurrentBalance]
                , [StatementBalance]
                , [OutStandingDeposits]
                , [OutStandingWithdrawals]
                , [PrevMonth1CurrentBalance]
                , [PrevMonth1StatementBalance]
                , [PrevMonth1OutStandingDeposits]
                , [PrevMonth1OutStandingWithdrawals]
                , [PrevMonth2CurrentBalance]
                , [PrevMonth2StatementBalance]
                , [PrevMonth2OutStandingDeposits]
                , [PrevMonth2OutStandingWithdrawals]
                , [CurrencyDescription]
                , [BuyExchangeRate]
                , [SellExchangeRate]
                , [BaseCcy]
                , [BaseCcyName]
                , [SecondConvertBuyRate]
                , [SecondConvertSellRate]
                , [SecondConvert]
                , [RateSell]
                , [RateBuy]
                )
                Select  [ab].[DatabaseName]
                      , [cn].[CompanyName]
                      , [ab].[Bank]
                      , [ab].[Description]
                      , [ab].[CashGlCode]
                      , [ab].[Currency]
                      , [ab].[CurrentBalance]
                      , [ab].[StatementBalance]
                      , [ab].[OutStandingDeposits]
                      , [ab].[OutStandingWithdrawals]
                      , [ab].[PrevMonth1CurrentBalance]
                      , [ab].[PrevMonth1StatementBalance]
                      , [ab].[PrevMonth1OutStandingDeposits]
                      , [ab].[PrevMonth1OutStandingWithdrawals]
                      , [ab].[PrevMonth2CurrentBalance]
                      , [ab].[PrevMonth2StatementBalance]
                      , [ab].[PrevMonth2OutStandingDeposits]
                      , [ab].[PrevMonth2OutStandingWithdrawals]
                      , [cc].[Description]
                      , [cc].[BuyExchangeRate]
                      , [cc].[SellExchangeRate]
                      , [cc].[BaseCcy]
                      , [cc].[BaseCcyName]
                      , [cc].[SecondConvertBuyRate]
                      , [cc].[SecondConvertSellRate]
                      , [cc].[SecondConvert]
                      , [cc].[RateSell]
                      , [cc].[RateBuy]
```

Author: Johnson, Chris

```
                From    [#ApBank] As [ab]
                        Left Join [#CurrencyConversions] As [cc] On [cc].[Currency]
= [ab].[Currency]
                                                                And [cc].[Database-
Name] = [ab].[DatabaseName]
                        Left Join [BlackBox].[Lookups].[CompanyNames] As [cn] On
[cn].[Company] = [ab].[DatabaseName];

--return results

        Print 8;
        Select  *
        From    [#Results]
        Order By [DatabaseName] Asc
              , [Bank]
              , [SecondConvert] Asc;

--tidy

        Drop Table [#Base];
        Drop Table [#SecondConvert];
        Drop Table [#TblCurrency];
        Drop Table [#CurrencyConversions];
        Drop Table [#ApBank];
    End;

GO
EXEC sp_addextendedproperty N'MS_Description', N'returns list of all bank balances',
'SCHEMA', N'Report', 'PROCEDURE', N'UspResults_AllBankBalances', NULL, NULL
GO
```

Uses

[Lookups].[CompanyNames]
[Process].[ExecForEachDB]
[Report]

Author: Johnson, Chris

[Report].[UspResults_AllPosAndReqs]

MS_Description

list of all purchase orders and requisitions

Parameters

Name	Data Type	Max Length (Bytes)
@Company	varchar(max)	max
@RedTagType	char	1
@RedTagUse	varchar(500)	500

SQL Script

```
CREATE Proc [Report].[UspResults_AllPosAndReqs]
    (
      @Company Varchar(Max)
    , @RedTagType Char(1)
    , @RedTagUse Varchar(500)
    )
As
    Begin
/*
Template designed by Chris Johnson, Prometic Group February 2016
*/
        If IsNumeric(@Company) = 0
            Begin
                Select  @Company = Upper(@Company);
            End;

--remove nocount on to speed up query

        Set NoCount On;

--Red tag

        Declare @RedTagDB Varchar(255)= Db_Name();
        Exec [Process].[UspInsert_RedTagLogs] @StoredProcDb = 'BlackBox' ,
            @StoredProcSchema = 'Report' ,
            @StoredProcName = 'UspResults_AllPosAndReqs' ,
            @UsedByType = @RedTagType , @UsedByName = @RedTagUse ,
            @UsedByDb = @RedTagDB;

--list the tables that are to be pulled back from each DB - if they are not found
the script will not be run against that db

        Declare @ListOfTables Varchar(Max) = 'AssetDepreciation,TblApTerms';

--create temporary tables to be pulled from different databases, including a column
to id
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_All-PosAndReqs

```
        Create Table [#PorMasterDetail]
            (
              [DatabaseName] Varchar(150)        collate latin1_general_bin
            , [MRequisition] Varchar(50)        collate latin1_general_bin
            , [PurchaseOrder] Varchar(50)        collate latin1_general_bin
            , [Line] Int
            , [MStockCode] Varchar(50)            collate latin1_general_bin
            , [MStockDes] Varchar(200)            collate latin1_general_bin
            , [MOrderQty] Numeric(20 , 8)
            , [MPriceUom] Varchar(10)            collate latin1_general_bin
            , [MLatestDueDate] Date
            , [MSupCatalogue] Varchar(50)        collate latin1_general_bin
            , [MPrice] Numeric(20 , 2)
            , [MForeignPrice] Numeric(20 , 2)
            , [MGlCode] Varchar(100)            collate latin1_general_bin
            , [MCompleteFlag] Char(1)            collate latin1_general_bin
            , [MProductClass] Varchar(20)        collate latin1_general_bin
            );
        Create Table [#ReqDetail]
            (
              [DatabaseName] Varchar(150)        collate latin1_general_bin
            , [Requisition] Varchar(50)            collate latin1_general_bin
            , [Line] Int
            , [StockCode] Varchar(50)            collate latin1_general_bin
            , [StockDescription] Varchar(200)    collate latin1_general_bin
            , [OrderQty] Numeric(20 , 8)
            , [OrderUom] Varchar(10)            collate latin1_general_bin
            , [DueDate] Date
            , [SupCatalogueNum] Varchar(50)        collate latin1_general_bin
            , [Price] Numeric(20 , 2)
            , [GlCode] Varchar(100)                collate latin1_general_bin
            , [Originator] Varchar(100)            collate latin1_general_bin
            , [Buyer] Varchar(100)                collate latin1_general_bin
            , [ReqnStatus] Varchar(150)            collate latin1_general_bin
            , [Operator] Varchar(150)            collate latin1_general_bin
            , [ApprovedDate] Date
            , [DateReqnRaised] Date
            , [DatePoConfirmed] Date
            , [ProductClass] Varchar(20)        collate latin1_general_bin
            , [Supplier] Varchar(30)            collate latin1_general_bin
            );
        Create Table [#PorMasterHdr]
            (
              [DatabaseName] Varchar(150)        collate latin1_general_bin
            , [PurchaseOrder] Varchar(50)        collate latin1_general_bin
            , [Buyer] Varchar(100)                collate latin1_general_bin
            , [DatePoCompleted] Date
            , [OrderStatus] Varchar(10)            collate latin1_general_bin
            , [Supplier] Varchar(30)            collate latin1_general_bin
            );
        Create Table [#ProjectList]
            (
              [DatabaseName] Varchar(150)        collate latin1_general_bin
            , [PurchaseOrder] Varchar(50)        collate latin1_general_bin
            , [Line] Int
            , [ProjectName] Varchar(255)        collate latin1_general_bin
            );
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_All-PosAndReqs

```
        Create Table [#ReqUser]
            (
              [DatabaseName] Varchar(150)        collate latin1_general_bin
            , [Originator] Varchar(100)            collate latin1_general_bin
            , [OriginatorName] Varchar(255)        collate latin1_general_bin
            );
        Create Table [#ApSupplier]
            (
              [DatabaseName] Varchar(150)        collate latin1_general_bin
            , [Supplier] Varchar(30)            collate latin1_general_bin
            , [SupplierName] Varchar(150)        collate latin1_general_bin
            );


--create script to pull data from each db into the tables

        Declare @SQLPorMasterDetail Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end'
            + --Only query DBs beginning SysProCompany

            '
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN'
            + --only companies selected in main run, or if companies selected then
all

            '
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL'' and IsNumeric(@DBCode)=1
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT'
            + --count number of tables requested (number of commas plus one)

            '
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')'
            + --Count of the tables requested how many exist in the db

            '
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'','')) '
            + --only if the count matches (all the tables exist in the requested db)
then run the script

            '
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
                    Insert [#PorMasterDetail]
            ( [DatabaseName]
            , [MRequisition]
            , [PurchaseOrder]
            , [Line]
            , [MStockCode]
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_All-PosAndReqs

```
            , [MStockDes]
            , [MOrderQty]
            , [MPriceUom]
            , [MLatestDueDate]
            , [MSupCatalogue]
            , [MPrice]
            , [MForeignPrice]
            , [MGlCode]
            , [MCompleteFlag]
            , [MProductClass]
            )
    SELECT [DatabaseName]=@DBCode
         , [PMD].[MRequisition]
         , [PMD].[PurchaseOrder]
         , [PMD].[Line]
         , [PMD].[MStockCode]
         , [PMD].[MStockDes]
         , [PMD].[MOrderQty]
         , [PMD].[MPriceUom]
         , [PMD].[MLatestDueDate]
         , [PMD].[MSupCatalogue]
         , [PMD].[MPrice]
         , [PMD].[MForeignPrice]
         , [PMD].[MGlCode]
         , [PMD].[MCompleteFlag]
         , [PMD].[MProductClass] FROM [PorMasterDetail] As [PMD]
            End
    End';
        Declare @SQLReqDetail Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end'
            + --Only query DBs beginning SysProCompany

            '
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN'
            + --only companies selected in main run, or if companies selected then
all

            '
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL'' and IsNumeric(@DBCode)=1
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT'
            + --count number of tables requested (number of commas plus one)

            '
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')'
            + --Count of the tables requested how many exist in the db

            '
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'','')) '
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_All-PosAndReqs

```
            + --only if the count matches (all the tables exist in the requested db)
then run the script

            '
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
                    Insert [#ReqDetail]
            ( [DatabaseName]
            , [Requisition]
            , [Line]
            , [StockCode]
            , [StockDescription]
            , [OrderQty]
            , [OrderUom]
            , [DueDate]
            , [SupCatalogueNum]
            , [Price]
            , [GlCode]
            , [Originator]
            , [Buyer]
            , [ReqnStatus]
            , [Operator]
            , [ApprovedDate]
            , [DateReqnRaised]
            , [DatePoConfirmed]
            , [ProductClass]
            , [Supplier]
            )
    SELECT [DatabaseName]=@DBCode
         , [RD].[Requisition]
         , [RD].[Line]
         , [RD].[StockCode]
         , [RD].[StockDescription]
         , [RD].[OrderQty]
         , [RD].[OrderUom]
         , [RD].[DueDate]
         , [RD].[SupCatalogueNum]
         , [RD].[Price]
         , [RD].[GlCode]
         , [RD].[Originator]
         , [RD].[Buyer]
         , [RD].[ReqnStatus]
         , [RD].[Operator]
         , [RD].[ApprovedDate]
         , [RD].[DateReqnRaised]
         , [RD].[DatePoConfirmed]
         , [RD].[ProductClass]
         , [RD].[Supplier] FROM [ReqDetail] As [RD]
            End
    End';
        Declare @SQLPorMasterHdr Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end'
            + --Only query DBs beginning SysProCompany
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_All-PosAndReqs

```
            '
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN'
            + --only companies selected in main run, or if companies selected then
all

            '
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL'' and IsNumeric(@DBCode)=1
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT'
            + --count number of tables requested (number of commas plus one)

            '
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')'
            + --Count of the tables requested how many exist in the db

            '
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'','')) '
            + --only if the count matches (all the tables exist in the requested db)
then run the script

            '
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
                    Insert [#PorMasterHdr]
            ( [DatabaseName]
            , [PurchaseOrder]
            , [Buyer]
            , [DatePoCompleted]
            , [OrderStatus]
            , [Supplier]
            )
    SELECT [DatabaseName]=@DBCode
         , [PMH].[PurchaseOrder]
         , [PMH].[Buyer]
         , [PMH].[DatePoCompleted]
         , [PMH].[OrderStatus]
         , [PMH].[Supplier] FROM [PorMasterHdr] As [PMH]
            End
    End';
        Declare @SQLProjectList Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end'
            + --Only query DBs beginning SysProCompany

            '
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN'
            + --only companies selected in main run, or if companies selected then
all
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_All-PosAndReqs

```
            '
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL'' and IsNumeric(@DBCode)=1
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT'
            + --count number of tables requested (number of commas plus one)

            '
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')'
            + --Count of the tables requested how many exist in the db

            '
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'','')) '
            + --only if the count matches (all the tables exist in the requested db)
then run the script

            '
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
            Insert [#ProjectList]
                    ( [DatabaseName]
                    , [PurchaseOrder]
                    , [Line]
                    , [ProjectName]
                    )
            Select  @DBCode
                  , [GD].[PurchaseOrder]
                  , Line = [GD].[PurchaseOrderLin]
                  , ProjectName = [GAC].[Description]
            From    [dbo].[GrnDetails] As [GD]
                    Left Join [dbo].[GenJournalDetail]
                    As [GJD] On [GJD].[Journal] = [GD].[Journal]
                    Left Join [dbo].[GenAnalysisTrn] As [GAT] On [GAT].[Analysis-
Entry] = [GJD].[AnalysisEntry]
                    Left Join [dbo].[GenAnalysisCode] As GAC On [GAT].[Analysis-
Code1] = [GAC].[AnalysisCode]
                                                                        And
[GAC].[AnalysisType] = 1
            Where   [GAC].[Description] Is Not Null
            Group By [GD].[PurchaseOrder]
                  , [GD].[PurchaseOrderLin]
                  , [GAC].[Description];
            End
    End';
        Declare @SQLReqUser Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end'
            + --Only query DBs beginning SysProCompany

            '
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN'
            + --only companies selected in main run, or if companies selected then
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_All-PosAndReqs

```
all

            '
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL'' and IsNumeric(@DBCode)=1
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT'
            + --count number of tables requested (number of commas plus one)

            '
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')'
            + --Count of the tables requested how many exist in the db

            '
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'','')) '
            + --only if the count matches (all the tables exist in the requested db)
then run the script

            '
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
            Insert [#ReqUser]
                    ( [DatabaseName],[Originator] , [OriginatorName] )
            SELECT @DBCode
                , [Originator]= [UserCode]
                , [OriginatorName]=[UserName]
            From [dbo].[ReqUser]
            End
    End';
        Declare @SQLApSupplier Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end'
            + --Only query DBs beginning SysProCompany

            '
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN'
            + --only companies selected in main run, or if companies selected then
all

            '
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL'' and IsNumeric(@DBCode)=1
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT'
            + --count number of tables requested (number of commas plus one)

            '
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')'
            + --Count of the tables requested how many exist in the db
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_All-PosAndReqs

```
            '
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'',')) '
            + --only if the count matches (all the tables exist in the requested db)
then run the script

            '
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
            Insert [#ApSupplier]
            ( [DatabaseName]
            , [Supplier]
            , [SupplierName]
            )
            SELECT [DatabaseName]=@DBCode
                 , [AS].[Supplier]
                 , [AS].[SupplierName] FROM  [ApSupplier] [AS]
            End
    End';
--Enable this function to check script changes (try to run script directly against
db manually)

--Print @SQL


--execute script against each db, populating the base tables

        Exec [Process].[ExecForEachDB] @cmd = @SQLPorMasterDetail;
        Exec [Process].[ExecForEachDB] @cmd = @SQLReqDetail;
        Exec [Process].[ExecForEachDB] @cmd = @SQLPorMasterHdr;
        Exec [Process].[ExecForEachDB] @cmd = @SQLProjectList;
        Exec [Process].[ExecForEachDB] @cmd = @SQLReqUser;
        Exec [Process].[ExecForEachDB] @cmd = @SQLApSupplier;



--define the results you want to return

        Create Table [#Results]
            (
              [DatabaseName] Varchar(150)            collate latin1_general_bin
            , [Requisition] Varchar(50)                collate latin1_general_bin
            , [PurchaseOrder] Varchar(50)            collate latin1_general_bin
            , [Line] Int
            , [StockCode] Varchar(50)                collate latin1_general_bin
            , [ProjectName] Varchar(50)                collate latin1_general_bin
            , [OriginalStockCode] Varchar(50)        collate latin1_general_bin
            , [StockDescription] Varchar(200)        collate latin1_general_bin
            , [OrderQty] Numeric(20 , 8)
            , [OrderUom] Varchar(10)                collate latin1_general_bin
            , [DueDate] Date
            , [SupCatalogueNum] Varchar(100)        collate latin1_general_bin
            , [Price] Numeric(20 , 2)
            , [MForeignPrice] Numeric(20 , 2)
            , [ProductClass] Varchar(250)            collate latin1_general_bin
            , [GlCode] Varchar(200)                    collate latin1_general_bin
```

Author: Johnson, Chris

```
            , [GlDescription] Varchar(200)            collate latin1_general_bin
            , [GlGroup] Varchar(20)                   collate latin1_general_bin
            , [Originator] Varchar(150)               collate latin1_general_bin
            , [OriginatorName] Varchar(255)           collate latin1_general_bin
            , [Buyer] Varchar(150)                    collate latin1_general_bin
            , [RequisitionStatus] Varchar(150)        collate latin1_general_bin
            , [Supplier] Varchar(100)                 collate latin1_general_bin
            , [DatePoConfirmed] Date
            , [DatePoCompleted] Date
            , [RequisitionOperator] Varchar(150)    collate latin1_general_bin
            , [ApprovedDate] Date
            , [DateRequisitionRaised] Date
            , [PurchaseOrders] Bit
            , [OpenRequisitions] Bit
            , [POStatus] Varchar(20)                collate latin1_general_bin
            , [Capex] Varchar(20)                   collate latin1_general_bin
            , [DeptCode] Char(5)                    collate latin1_general_bin
            , [OrderStatus] Varchar(150)            collate latin1_general_bin
            , [OrderStatusDescription] Varchar(250)    collate latin1_general_bin
            , [MCompleteFlag] Char(1)               collate latin1_general_bin
            , [CompanyName] Varchar(200)            collate latin1_general_bin
            , [ShortName] Varchar(250)                collate latin1_general_bin
            , [Company] Varchar(150)                collate latin1_general_bin
            , [Currency] Varchar(10)                collate latin1_general_bin
            , [CADDivision] Float
            , [CADMultiply] Float
            , [StartDateTime] DateTime2
            , [SupplierName] Varchar(150)            collate latin1_general_bin
            );

--Placeholder to create indexes as required


--script to combine base data and insert into results table

        Insert  [#Results]
                ( [DatabaseName]
                , [Requisition]
                , [PurchaseOrder]
                , [Line]
                , [StockCode]
                , [ProjectName]
                , [OriginalStockCode]
                , [StockDescription]
                , [OrderQty]
                , [OrderUom]
                , [DueDate]
                , [SupCatalogueNum]
                , [Price]
                , [MForeignPrice]
                , [ProductClass]
                , [GlCode]
                , [GlDescription]
                , [GlGroup]
                , [Originator]
                , [OriginatorName]
                , [Buyer]
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_All-PosAndReqs

```
                , [RequisitionStatus]
                , [Supplier]
                , [DatePoConfirmed]
                , [DatePoCompleted]
                , [RequisitionOperator]
                , [ApprovedDate]
                , [DateRequisitionRaised]
                , [PurchaseOrders]
                , [OpenRequisitions]
                , [POStatus]
                , [Capex]
                , [DeptCode]
                , [OrderStatus]
                , [OrderStatusDescription]
                , [MCompleteFlag]
                , [CompanyName]
                , [ShortName]
                , [Company]
                , [Currency]
                , [CADDivision]
                , [CADMultiply]
                , [StartDateTime]
                , [SupplierName]
                )
                Select  [DatabaseName] = Coalesce([RD].[DatabaseName] ,
                                                  [PMD].[DatabaseName])
                      , [Requisition] = Coalesce(Case When [PMD].[MRequisition] = ''
                                                      Then Null
                                                      Else [PMD].[MRequisition]
                                                 End , [RD].[Requisition])
                      , [PMD].[PurchaseOrder]
                      , [Line] = Coalesce([PMD].[Line] , [RD].[Line])
                      , [StockCode] = Case When [PC].[ProductClassDescription] =
'Raw Materials'
                                           Then Coalesce([PMD].[MStockCode] ,
                                                         [RD].[StockCode])
                                           Else Null
                                      End
                      , [ProjectName] = Case When [PMD].[DatabaseName] <> '10'
                                             Then [PL].[ProjectName]
                                             When
Replace(Lower(Coalesce([PMD].[MStockCode] ,
                                                                    [RD].[StockCode])) ,
                                                                ' ' , '') In (
                                                  'area51' , 'itproject' )
                                             Then Coalesce([PMD].[MStockCode] ,
                                                           [RD].[StockCode])
                                             Else Null
                                        End
                      , [OriginalStockCode] = Coalesce([PMD].[MStockCode] ,
                                                       [RD].[StockCode])
                      , [StockDescription] = Coalesce([PMD].[MStockDes] ,
                                                      [RD].[StockDescription])
                      , [OrderQty] = Coalesce([PMD].[MOrderQty] ,
                                              [RD].[OrderQty])
                      , [OrderUom] = Lower(Coalesce([PMD].[MPriceUom] ,
                                                    [RD].[OrderUom]))
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_All-PosAndReqs

```
                        , [DueDate] = Convert(Date , Coalesce([PMD].[MLatestDueDate] ,
                                                              [RD].[DueDate]))
                        , [SupCatalogueNum] = Coalesce(Case When [PMD].[MSupCatalogue]
= ''
                                                            Then Null
                                                            Else [PMD].[MSupCatalogue]
                                                       End ,
                                                       Case When [RD].[SupCatalogue-
Num] = ''
                                                            Then Null
                                                            Else [RD].[SupCatalogue-
Num]
                                                       End)
                        , [Price] = Coalesce([PMD].[MPrice] , [RD].[Price])
                        , [PMD].[MForeignPrice]
                        , [ProductClass] = [PC].[ProductClassDescription]
                        , [GlCode] = Coalesce(Case When [PMD].[MGlCode] = ''
                                                   Then Null
                                                   Else [PMD].[MGlCode]
                                              End ,
                                              Case When [RD].[GlCode] = ''
                                                   Then Null
                                                   Else [RD].[GlCode]
                                              End)
                        , [GlDescription] = [GM].[Description]
                        , [GM].[GlGroup]
                        , [RD].[Originator]
                        , [OriginatorName] = [RU].[OriginatorName]
                        , [Buyer] = Coalesce(Case When [PMH].[Buyer] = ''
                                                  Then Null
                                                  Else [PMH].[Buyer]
                                             End ,
                                             Case When [RD].[Buyer] = ''
                                                  Then Null
                                                  Else [RD].[Buyer]
                                             End)
                        , [RequisitionStatus] = [RS].[ReqnStatusDescription]
                        , [Supplier] = Coalesce([PMH].[Supplier] ,
                                                [RD].[Supplier])
                        , [DatePoConfirmed] = Convert(Date , [RD].[DatePoConfirmed])
                        , [DatePoCompleted] = Convert(Date , [PMH].[DatePoCompleted])
                        , [RequisitionOperator] = [RD].[Operator]
                        , [ApprovedDate] = Convert(Date , [RD].[ApprovedDate])
                        , [DateRequisitionRaised] = Convert(Date , [RD].[DateReqn-
Raised])
                        , [PurchaseOrders] = Case When [PMD].[PurchaseOrder] Is Null
                                                  Then 0
                                                  Else 1
                                             End
                        , [OpenRequisitions] = Case When [PMD].[PurchaseOrder] Is Null
                                                    Then 1
                                                    Else 0
                                               End
                        , [POStatus] = Case When [PMH].[DatePoCompleted] Is Null
                                            Then 'Open'
                                            Else 'Closed'
                                       End
                        , [Capex] = Case When Substring([GM].[GlCode] , 5 , 5) =
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_All-PosAndReqs

```
'22999'
                                    Then 'Capex'
                                    Else Null
                                End
                    , [DeptCode] = Substring([GM].[GlCode] , 5 , 5)
                    , [PMH].[OrderStatus]
                    , [POS].[OrderStatusDescription]
                    , [PMD].[MCompleteFlag]
                    , [CN].[CompanyName]
                    , [CN].[ShortName]
                    , [Company] = Coalesce([PMH].[DatabaseName] ,
                                           [RD].[DatabaseName])
                    , [CR].[Currency]
                    , [CR].[CADDivision]
                    , [CR].[CADMultiply]
                    , [CR].[StartDateTime]
                    , [AS].[SupplierName]
                From    [#PorMasterDetail] As [PMD]
                        Full Outer Join [#ReqDetail] As [RD]
                            On [RD].[Requisition] = [PMD].[MRequisition]
                               And [RD].[Line] = [PMD].[Line]
                               And [RD].[DatabaseName] = [PMD].[DatabaseName]
                        Left Join [#PorMasterHdr] As [PMH]
                            On [PMH].[PurchaseOrder] = [PMD].[PurchaseOrder]
                               And [PMH].[DatabaseName] = [PMD].[DatabaseName]
                        Left Join [SysproCompany40].[dbo].[GenMaster] As [GM]
                            On [GM].[GlCode] = Coalesce([PMD].[MGlCode] ,
                                                        [RD].[GlCode])
                        Left Join [BlackBox].[Lookups].[ProductClass] As [PC]
                            On [PC].[ProductClass] = Coalesce([PMD].[MProductClass]
,
                                                              [RD].[ProductClass])
                               And [PC].[Company] = Coalesce([PMD].[DatabaseName] ,
                                                             [RD].[DatabaseName])
                        Left Join [BlackBox].[Lookups].[PurchaseOrderStatus]
                            As [POS]
                            On [POS].[OrderStatusCode] = [PMH].[OrderStatus]
                               And [POS].[Company] = [PMH].[DatabaseName]
                        Left Join [BlackBox].[Lookups].[CompanyNames] As [CN]
                            On [CN].[Company] = Coalesce([PMH].[DatabaseName] ,
                                                         [RD].[DatabaseName])
                        Left Join [BlackBox].[Lookups].[CurrencyRates] As [CR]
                            On [CR].[Currency] = [CN].[Currency]
                               And GetDate() Between [CR].[StartDateTime]
                                             And     [CR].[EndDateTime]
                        Left Join [#ProjectList] As [PL]
                            On [PL].[Line] = [PMD].[Line]
                               And [PL].[PurchaseOrder] = [PMD].[PurchaseOrder]
                               And [PL].[DatabaseName] = [PMD].[DatabaseName]
                        Left Join [#ReqUser] As [RU]
                            On [RU].[Originator] = [RD].[Originator]
                               And [RU].[DatabaseName] = [RD].[DatabaseName]
                        Left Join [Lookups].[ReqnStatus] As [RS]
                            On [RS].[ReqnStatusCode] = [RD].[ReqnStatus]
                               And [RS].[Company] = [RD].[DatabaseName]
                        Left Join [#ApSupplier] [AS]
                            On [AS].[Supplier] = Coalesce([PMH].[Supplier] ,
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_All-PosAndReqs

```
                                                                [RD].[Supplier])
                                    And [AS].[DatabaseName] = Coalesce([PMH].[Database-
Name] ,
                                                                    [RD].[DatabaseName])
                    Where   Coalesce([PMD].[MPrice] , [RD].[Price]) <> 0;

--return results

            Select  [Requisition] = Case When IsNumeric([Requisition]) = 1
                                         Then Convert(Varchar(50) , Convert(Int ,
[Requisition]))
                                         Else Null
                                    End
                  , [PurchaseOrder] = Case When IsNumeric([PurchaseOrder]) = 1
                                           Then Convert(Varchar(50) , Convert(Int ,
[PurchaseOrder]))
                                           Else Null
                                      End
                  , [Line]
                  , [StockCode]
                  , [ProjectName]
                  , [OriginalStockCode]
                  , [StockDescription]
                  , [OrderQty]
                  , [OrderUom]
                  , [DueDate]
                  , [SupCatalogueNum]
                  , [Price]
                  , [MForeignPrice]
                  , [ProductClass]
                  , [GlCode]
                  , [CompanyGlCode] = [DatabaseName] + '.' + [GlCode]
                  , [GlDescription]
                  , [GlGroup]
                  , [Originator]
                  , [OriginatorName]
                  , [Buyer]
                  , [RequisitionStatus]
                  , [Supplier]
                  , [DatePoConfirmed]
                  , [DatePoCompleted]
                  , [RequisitionOperator]
                  , [ApprovedDate]
                  , [DateRequisitionRaised]
                  , [PurchaseOrders]
                  , [OpenRequisitions]
                  , [POStatus]
                  , [Capex]
                  , [DeptCode]
                  , [OrderStatus]
                  , [OrderStatusDescription]
                  , [MCompleteFlag]
                  , [CompanyName]
                  , [Company]
                  , [Currency]
                  , [CADDivision]
                  , [CADMultiply]
                  , [StartDateTime]
```

Author: Johnson, Chris

```
              , [ShortName]
              , [SupplierName]
        From    [#Results]
        Where   Coalesce([RequisitionStatus] , '') <> 'Cancelled'
        Order By [PurchaseOrder] Asc
              , [Line] Asc;
    End;

GO
EXEC sp_addextendedproperty N'MS_Description', N'list of all purchase orders and
requisitions', 'SCHEMA', N'Report', 'PROCEDURE', N'UspResults_AllPosAndReqs', NULL,
NULL
GO
```

Uses

[Lookups].[CompanyNames]
[Lookups].[CurrencyRates]
[Lookups].[ProductClass]
[Lookups].[PurchaseOrderStatus]
[Lookups].[ReqnStatus]
[Process].[ExecForEachDB]
[Process].[UspInsert_RedTagLogs]
[Report]

[Report].[UspResults_AllSalesOrders]

MS_Description

list of all sales orders

Parameters

Name	Data Type	Max Length (Bytes)
@Company	varchar(max)	max
@RedTagType	char	1
@RedTagUse	varchar(500)	500

SQL Script

```
CREATE Proc [Report].[UspResults_AllSalesOrders]
    (
      @Company Varchar(Max)
    , @RedTagType Char(1)
    , @RedTagUse Varchar(500)
    )
As
    Begin
/*
Template designed by Chris Johnson, Prometic Group September 2015
Stored procedure set out to query multiple databases with the same information and
return it in a collated format
*/
        Set NoCount On;

        If IsNumeric(@Company) = 0
            Begin
                Select  @Company = Upper(@Company);
            End;

--Red tag

        Declare @RedTagDB Varchar(255)= Db_Name();
        Exec [Process].[UspInsert_RedTagLogs] @StoredProcDb = 'BlackBox' ,
            @StoredProcSchema = 'Report' ,
            @StoredProcName = 'UspResults_Template' ,
            @UsedByType = @RedTagType , @UsedByName = @RedTagUse ,
            @UsedByDb = @RedTagDB;

--list the tables that are to be pulled back from each DB - if they are not found
the script will not be run against that db

        Declare @ListOfTables Varchar(Max) = 'SorMaster,SorDetail,ArCustomer';

--create temporary tables to be pulled from different databases, including a column
to id
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_All-SalesOrders

```
        Create Table [#ArCustomer]
            (
              [DatabaseName] sysname            collate latin1_general_bin
            , [Customer] Varchar(15)            collate latin1_general_bin
            , [Name] Varchar(50)                collate latin1_general_bin
            , [ShortName] Varchar(20)            collate latin1_general_bin
            , [Area] Varchar(10)                collate latin1_general_bin
            , [Currency] Varchar(3)                collate latin1_general_bin
            , [DateCustAdded] Date
            , [Nationality] Char(3)                collate latin1_general_bin
            , [SoldToAddr1] Varchar(40)            collate latin1_general_bin
            , [SoldToAddr2] Varchar(40)            collate latin1_general_bin
            , [SoldToAddr3] Varchar(40)            collate latin1_general_bin
            , [SoldToAddr3Loc] Varchar(40)        collate latin1_general_bin
            , [SoldToAddr4] Varchar(40)            collate latin1_general_bin
            , [SoldToAddr5] Varchar(40)            collate latin1_general_bin
            , [SoldPostalCode] Varchar(10)        collate latin1_general_bin
            );
        Create Table [#SorDetail]
            (
              [DatabaseName] sysname            collate latin1_general_bin
            , [SalesOrder] Varchar(20)            collate latin1_general_bin
            , [SalesOrderLine] Int
            , [LineType] Char(1)                collate latin1_general_bin
            , [MStockCode] Varchar(30)            collate latin1_general_bin
            , [MStockDes] Varchar(50)            collate latin1_general_bin
            , [MWarehouse] Varchar(10)            collate latin1_general_bin
            , [MBin] Varchar(20)                collate latin1_general_bin
            , [MOrderQty] Numeric(20 , 6)
            , [MShipQty] Numeric(20 , 6)
            , [MBackOrderQty] Numeric(20 , 6)
            , [MUnitCost] Numeric(20 , 6)
            , [MDecimals] Int
            , [MOrderUom] Varchar(10)            collate latin1_general_bin
            , [MStockQtyToShp] Numeric(20 , 6)
            , [MStockingUom] Varchar(10)        collate latin1_general_bin
            , [MPrice] Numeric(20 , 6)
            , [MPriceUom] Varchar(10)            collate latin1_general_bin
            , [MProductClass] Varchar(20)        collate latin1_general_bin
            , [MTaxCode] Char(3)                collate latin1_general_bin
            , [MLineShipDate] Date
            , [MQtyDispatched] Numeric(20 , 6)
            , [QtyReserved] Numeric(20 , 6)
            , [NComment] Varchar(100)            collate latin1_general_bin
            );
        Create Table [#SorMaster]
            (
              [DatabaseName] sysname            collate latin1_general_bin
            , [SalesOrder] Varchar(20)            collate latin1_general_bin
            , [OrderStatus] Char(1)                collate latin1_general_bin
            , [CustomerPoNumber] Varchar(30)    collate latin1_general_bin
            , [OrderDate] Date
            , [EntrySystemDate] Date
            , [ReqShipDate] Date
            , [ShippingInstrs] Varchar(50)        collate latin1_general_bin
            , [ExchangeRate] Float
            , [MulDiv] Char(1)                    collate latin1_general_bin
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_All-SalesOrders

```
            , [Currency] Char(3)                collate latin1_general_bin
            , [ShipAddress1] Varchar(40)        collate latin1_general_bin
            , [ShipAddress2] Varchar(40)        collate latin1_general_bin
            , [ShipAddress3] Varchar(40)        collate latin1_general_bin
            , [ShipAddress3Loc] Varchar(40)        collate latin1_general_bin
            , [ShipAddress4] Varchar(40)        collate latin1_general_bin
            , [ShipAddress5] Varchar(40)        collate latin1_general_bin
            , [ShipPostalCode] Varchar(10)        collate latin1_general_bin
            , [Customer] Varchar(15)            collate latin1_general_bin
            , [CancelledFlag] Char(1)            collate latin1_general_bin
            );

--create script to pull data from each db into the tables

        Declare @SQLSorCusts Varchar(Max) = 'USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            BEGIN
                Insert [#ArCustomer]
                        ( [DatabaseName]
                        , [Customer]
                        , [Name]
                        , [ShortName]
                        , [Area]
                        , [Currency]
                        , [DateCustAdded]
                        , [Nationality]
                        , [SoldToAddr1]
                        , [SoldToAddr2]
                        , [SoldToAddr3]
                        , [SoldToAddr3Loc]
                        , [SoldToAddr4]
                        , [SoldToAddr5]
                        , [SoldPostalCode]
                        )
                SELECT [DatabaseName]=@DBCode
                     , [AC].[Customer]
                     , [AC].[Name]
                     , [AC].[ShortName]
                     , [AC].[Area]
                     , [AC].[Currency]
                     , [AC].[DateCustAdded]
                     , [AC].[Nationality]
                     , [AC].[SoldToAddr1]
                     , [AC].[SoldToAddr2]
                     , [AC].[SoldToAddr3]
                     , [AC].[SoldToAddr3Loc]
                     , [AC].[SoldToAddr4]
                     , [AC].[SoldToAddr5]
                     , [AC].[SoldPostalCode] FROM [ArCustomer] [AC]

                Insert [#SorDetail]
```

Author: Johnson, Chris

```
                        ( [DatabaseName]
                        , [SalesOrder]
                        , [SalesOrderLine]
                        , [LineType]
                        , [MStockCode]
                        , [MStockDes]
                        , [MWarehouse]
                        , [MBin]
                        , [MOrderQty]
                        , [MShipQty]
                        , [MBackOrderQty]
                        , [MUnitCost]
                        , [MDecimals]
                        , [MOrderUom]
                        , [MStockQtyToShp]
                        , [MStockingUom]
                        , [MPrice]
                        , [MPriceUom]
                        , [MProductClass]
                        , [MTaxCode]
                        , [MLineShipDate]
                        , [MQtyDispatched]
                        , [QtyReserved]
                        , [NComment]
                        )
                SELECT [DatabaseName]=@DBCode
                     , [SD].[SalesOrder]
                     , [SD].[SalesOrderLine]
                     , [SD].[LineType]
                     , [SD].[MStockCode]
                     , [SD].[MStockDes]
                     , [SD].[MWarehouse]
                     , [SD].[MBin]
                     , [SD].[MOrderQty]
                     , [SD].[MShipQty]
                     , [SD].[MBackOrderQty]
                     , [SD].[MUnitCost]
                     , [SD].[MDecimals]
                     , [SD].[MOrderUom]
                     , [SD].[MStockQtyToShp]
                     , [SD].[MStockingUom]
                     , [SD].[MPrice]
                     , [SD].[MPriceUom]
                     , [SD].[MProductClass]
                     , [SD].[MTaxCode]
                     , [SD].[MLineShipDate]
                     , [SD].[MQtyDispatched]
                     , [SD].[QtyReserved]
                     , [SD].[NComment] FROM [SorDetail] [SD]

                Insert [#SorMaster]
                        ( [DatabaseName]
                        , [SalesOrder]
                        , [OrderStatus]
                        , [CustomerPoNumber]
                        , [OrderDate]
                        , [EntrySystemDate]
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_All-SalesOrders

```
                        , [ReqShipDate]
                        , [ShippingInstrs]
                        , [ExchangeRate]
                        , [MulDiv]
                        , [Currency]
                        , [ShipAddress1]
                        , [ShipAddress2]
                        , [ShipAddress3]
                        , [ShipAddress3Loc]
                        , [ShipAddress4]
                        , [ShipAddress5]
                        , [ShipPostalCode]
                        , [Customer]
                        , [CancelledFlag]
                        )
                SELECT [DatabaseName]=@DBCode
                     , [SM].[SalesOrder]
                     , [SM].[OrderStatus]
                     , [SM].[CustomerPoNumber]
                     , [SM].[OrderDate]
                     , [SM].[EntrySystemDate]
                     , [SM].[ReqShipDate]
                     , [SM].[ShippingInstrs]
                     , [SM].[ExchangeRate]
                     , [SM].[MulDiv]
                     , [SM].[Currency]
                     , [SM].[ShipAddress1]
                     , [SM].[ShipAddress2]
                     , [SM].[ShipAddress3]
                     , [SM].[ShipAddress3Loc]
                     , [SM].[ShipAddress4]
                     , [SM].[ShipAddress5]
                     , [SM].[ShipPostalCode]
                     , [SM].[Customer]
                     , [SM].[CancelledFlag] FROM [SorMaster] [SM]
            End
    End';

--Enable this function to check script changes (try to run script directly against
db manually)

--Print @SQL


--execute script against each db, populating the base tables

        Exec [Process].[ExecForEachDB_WithTableCheck] @cmd = @SQLSorCusts ,
            @SchemaTablesToCheck = @ListOfTables;


--define the results you want to return

        Create Table [#Results]
            (
              [DatabaseName] Varchar(150)        collate latin1_general_bin
            , [SalesOrder] Varchar(20)            collate latin1_general_bin
            , [OrderStatus] Varchar(150)        collate latin1_general_bin
```

Author: Johnson, Chris

```
            , [CustomerPoNumber] Varchar(30)    collate latin1_general_bin
            , [OrderDate] Date
            , [EntrySystemDate] Date
            , [ReqShipDate] Date
            , [ShippingInstrs] Varchar(50)        collate latin1_general_bin
            , [ExchangeRate] Float
            , [MulDiv] Char(1)                    collate latin1_general_bin
            , [Currency] Char(3)                collate latin1_general_bin
            , [ShipAddress1] Varchar(40)        collate latin1_general_bin
            , [ShipAddress2] Varchar(40)        collate latin1_general_bin
            , [ShipAddress3] Varchar(40)        collate latin1_general_bin
            , [ShipAddress3Loc] Varchar(40)        collate latin1_general_bin
            , [ShipAddress4] Varchar(40)        collate latin1_general_bin
            , [ShipAddress5] Varchar(40)        collate latin1_general_bin
            , [ShipPostalCode] Varchar(10)        collate latin1_general_bin
            , [SalesOrderLine] Int
            , [LineType] Int
            , [StockCode] Varchar(30)            collate latin1_general_bin
            , [StockDescription] Varchar(50)    collate latin1_general_bin
            , [Warehouse] Varchar(10)            collate latin1_general_bin
            , [Bin] Varchar(20)                    collate latin1_general_bin
            , [OrderQty] Numeric(20 , 6)
            , [ShipQty] Numeric(20 , 6)
            , [BackOrderQty] Numeric(20 , 6)
            , [UnitCost] Numeric(20 , 6)
            , [Decimals] Int
            , [OrderUom] Char(3)                collate latin1_general_bin
            , [StockQtyToShp] Numeric(20 , 6)
            , [StockingUom] Char(3)                collate latin1_general_bin
            , [Price] Numeric(20 , 6)
            , [PriceUom] Char(3)                collate latin1_general_bin
            , [ProductClass] Varchar(20)        collate latin1_general_bin
            , [TaxCode] Char(3)                    collate latin1_general_bin
            , [LineShipDate] Date
            , [QtyDispatched] Numeric(20 , 6)
            , [QtyReserved] Numeric(20 , 6)
            , [LineComment] Varchar(100)        collate latin1_general_bin
            , [Customer] Varchar(15)            collate latin1_general_bin
            , [CustomerName] Varchar(50)        collate latin1_general_bin
            , [CustomerShortName] Varchar(20)    collate latin1_general_bin
            , [Area] Varchar(10)                collate latin1_general_bin
            , [CustomerCurrency] Varchar(3)        collate latin1_general_bin
            , [DateCustAdded] Date
            , [Nationality] Char(3)                collate latin1_general_bin
            , [SoldToAddr1] Varchar(40)            collate latin1_general_bin
            , [SoldToAddr2] Varchar(40)            collate latin1_general_bin
            , [SoldToAddr3] Varchar(40)            collate latin1_general_bin
            , [SoldToAddr3Loc] Varchar(40)        collate latin1_general_bin
            , [SoldToAddr4] Varchar(40)            collate latin1_general_bin
            , [SoldToAddr5] Varchar(40)            collate latin1_general_bin
            , [SoldPostalCode] Varchar(10)        collate latin1_general_bin
            );

--Placeholder to create indexes as required


--script to combine base data and insert into results table
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_All-SalesOrders

```
        Insert  [#Results]
                ( [DatabaseName]
                , [SalesOrder]
                , [OrderStatus]
                , [CustomerPoNumber]
                , [OrderDate]
                , [EntrySystemDate]
                , [ReqShipDate]
                , [ShippingInstrs]
                , [ExchangeRate]
                , [MulDiv]
                , [Currency]
                , [ShipAddress1]
                , [ShipAddress2]
                , [ShipAddress3]
                , [ShipAddress3Loc]
                , [ShipAddress4]
                , [ShipAddress5]
                , [ShipPostalCode]
                , [SalesOrderLine]
                , [LineType]
                , [StockCode]
                , [StockDescription]
                , [Warehouse]
                , [Bin]
                , [OrderQty]
                , [ShipQty]
                , [BackOrderQty]
                , [UnitCost]
                , [Decimals]
                , [OrderUom]
                , [StockQtyToShp]
                , [StockingUom]
                , [Price]
                , [PriceUom]
                , [ProductClass]
                , [TaxCode]
                , [LineShipDate]
                , [QtyDispatched]
                , [QtyReserved]
                , [LineComment]
                , [Customer]
                , [CustomerName]
                , [CustomerShortName]
                , [Area]
                , [CustomerCurrency]
                , [DateCustAdded]
                , [Nationality]
                , [SoldToAddr1]
                , [SoldToAddr2]
                , [SoldToAddr3]
                , [SoldToAddr3Loc]
                , [SoldToAddr4]
                , [SoldToAddr5]
                , [SoldPostalCode]
                )
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_All-SalesOrders

```
Select  [SM].[DatabaseName]
      , [SM].[SalesOrder]
      , [OrderStatus] = [SOS].[OrderStatusDescription]
      , [SM].[CustomerPoNumber]
      , [SM].[OrderDate]
      , [SM].[EntrySystemDate]
      , [SM].[ReqShipDate]
      , [SM].[ShippingInstrs]
      , [SM].[ExchangeRate]
      , [SM].[MulDiv]
      , [SM].[Currency]
      , [SM].[ShipAddress1]
      , [SM].[ShipAddress2]
      , [SM].[ShipAddress3]
      , [SM].[ShipAddress3Loc]
      , [SM].[ShipAddress4]
      , [SM].[ShipAddress5]
      , [SM].[ShipPostalCode]
      , [SD].[SalesOrderLine]
      , [SD].[LineType]
      , [SD].[MStockCode]
      , [SD].[MStockDes]
      , [SD].[MWarehouse]
      , [SD].[MBin]
      , [SD].[MOrderQty]
      , [SD].[MShipQty]
      , [SD].[MBackOrderQty]
      , [SD].[MUnitCost]
      , [SD].[MDecimals]
      , [SD].[MOrderUom]
      , [SD].[MStockQtyToShp]
      , [SD].[MStockingUom]
      , [SD].[MPrice]
      , [SD].[MPriceUom]
      , [SD].[MProductClass]
      , [SD].[MTaxCode]
      , [SD].[MLineShipDate]
      , [SD].[MQtyDispatched]
      , [SD].[QtyReserved]
      , [SD].[NComment]
      , [AC].[Customer]
      , [AC].[Name]
      , [AC].[ShortName]
      , [AC].[Area]
      , [AC].[Currency]
      , [AC].[DateCustAdded]
      , [AC].[Nationality]
      , [AC].[SoldToAddr1]
      , [AC].[SoldToAddr2]
      , [AC].[SoldToAddr3]
      , [AC].[SoldToAddr3Loc]
      , [AC].[SoldToAddr4]
      , [AC].[SoldToAddr5]
      , [AC].[SoldPostalCode]
From    [#SorMaster] [SM]
        Left Join [#SorDetail] [SD]
            On [SD].[SalesOrder] = [SM].[SalesOrder]
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_All-SalesOrders

```
                                And [SD].[DatabaseName] = [SM].[DatabaseName]
                        Left Join [#ArCustomer] [AC]
                            On [AC].[Customer] = [SM].[Customer]
                                And [AC].[DatabaseName] = [SM].[DatabaseName]
                        Left Join [BlackBox].[Lookups].[SalesOrderStatus] [SOS]
                            On [SOS].[Company] = [SM].[DatabaseName]
                                And [SOS].[OrderStatusCode] = [SM].[OrderStatus]
                Where   Coalesce([SM].[CancelledFlag] , 'N') <> 'Y';


        Set NoCount Off;
--return results

        Select  [DatabaseName]
              , [SalesOrder]
              , [OrderStatus]
              , [CustomerPoNumber]
              , [OrderDate]
              , [EntrySystemDate]
              , [ReqShipDate]
              , [ShippingInstrs]
              , [ExchangeRate]
              , [MulDiv]
              , [Currency]
              , [ShipAddress1]
              , [ShipAddress2]
              , [ShipAddress3]
              , [ShipAddress3Loc]
              , [ShipAddress4]
              , [ShipAddress5]
              , [ShipPostalCode]
              , [SalesOrderLine]
              , [LineType]
              , [StockCode]
              , [StockDescription]
              , [Warehouse]
              , [Bin]
              , [OrderQty]
              , [ShipQty]
              , [BackOrderQty]
              , [UnitCost]
              , [Decimals]
              , [OrderUom]
              , [StockQtyToShp]
              , [StockingUom]
              , [Price]
              , [PriceUom]
              , [ProductClass]
              , [TaxCode]
              , [LineShipDate]
              , [QtyDispatched]
              , [QtyReserved]
              , [LineComment]
              , [Customer]
              , [CustomerName]
              , [CustomerShortName]
              , [Area]
```

Author: Johnson, Chris

```
              , [CustomerCurrency]
              , [DateCustAdded]
              , [Nationality]
              , [SoldToAddr1]
              , [SoldToAddr2]
              , [SoldToAddr3]
              , [SoldToAddr3Loc]
              , [SoldToAddr4]
              , [SoldToAddr5]
              , [SoldPostalCode]
        From    [#Results];

    End;

GO
EXEC sp_addextendedproperty N'MS_Description', N'list of all sales orders',
'SCHEMA', N'Report', 'PROCEDURE', N'UspResults_AllSalesOrders', NULL, NULL
GO
```

Uses

[Lookups].[SalesOrderStatus]
[Process].[ExecForEachDB_WithTableCheck]
[Process].[UspInsert_RedTagLogs]
[Report]

Author: Johnson, Chris

[Report].[UspResults_AllSuppliers]

MS_Description

list of all suppliers

Parameters

Name	Data Type	Max Length (Bytes)
@Company	varchar(max)	max
@RedTagType	char	1
@RedTagUse	varchar(500)	500

SQL Script

```
CREATE Proc [Report].[UspResults_AllSuppliers]
    (
      @Company Varchar(Max)
    , @RedTagType Char(1)
    , @RedTagUse Varchar(500)
    )
As
    Begin
/*
Template designed by Chris Johnson, Prometic Group September 2015
Stored procedure set out to query multiple databases with the same information and
return it in a collated format
*/
        Set NoCount On;

        If IsNumeric(@Company) = 0
            Begin
                Select  @Company = Upper(@Company);
            End;

--Red tag

        Declare @RedTagDB Varchar(255)= Db_Name();
        Exec [Process].[UspInsert_RedTagLogs] @StoredProcDb = 'BlackBox' ,
            @StoredProcSchema = 'Report' ,
            @StoredProcName = 'UspResults_AllSuppliers' ,
            @UsedByType = @RedTagType , @UsedByName = @RedTagUse ,
            @UsedByDb = @RedTagDB;

--list the tables that are to be pulled back from each DB - if they are not found
the script will not be run against that db

        Declare @ListOfTables Varchar(Max) = 'ApSupplier';

--create temporary tables to be pulled from different databases, including a column
to id
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_All-Suppliers

```
        Create Table [#ApSupplier]
            (
              [DatabaseName] sysname        collate Latin1_General_BIN
            , [Supplier] Varchar(15)        collate Latin1_General_BIN
            , [SupplierName] Varchar(50)    collate Latin1_General_BIN
            , [SupShortName] Varchar(20)    collate Latin1_General_BIN
            , [Branch] Varchar(10)            collate Latin1_General_BIN
            );



--create script to pull data from each db into the tables

        Declare @SQL Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            BEGIN
                Insert  [#ApSupplier]
        ( [DatabaseName]
        , [Supplier]
        , [SupplierName]
        , [SupShortName]
        , [Branch]
        )
        Select  @DBCode
              , [ApS].[Supplier]
              , [ApS].[SupplierName]
              , [ApS].[SupShortName]
              , [ApS].[Branch]
        From    [dbo].[ApSupplier] [ApS];
            End
    End';

--Enable this function to check script changes (try to run script directly against
db manually)

--Print @SQL


--execute script against each db, populating the base tables

        Exec [Process].[ExecForEachDB_WithTableCheck] @cmd = @SQL ,
            @SchemaTablesToCheck = @ListOfTables;


--define the results you want to return


--Placeholder to create indexes as required
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_All-Suppliers

```
--script to combine base data and insert into results table



        Set NoCount Off;
--return results

        Select  [ApS].[DatabaseName]
              , [ApS].[Supplier]
              , [ApS].[SupplierName]
              , [ApS].[SupShortName]
              , [ApS].[Branch]
        From    [#ApSupplier] [ApS];

    End;

GO
EXEC sp_addextendedproperty N'MS_Description', N'list of all suppliers', 'SCHEMA',
N'Report', 'PROCEDURE', N'UspResults_AllSuppliers', NULL, NULL
GO
```

Uses

[Process].[ExecForEachDB_WithTableCheck]
[Process].[UspInsert_RedTagLogs]
[Report]

Author: Johnson, Chris

[Report].[UspResults_AmendmentJnl]

MS_Description

details from the amendment jnl

Parameters

Name	Data Type	Max Length (Bytes)
@Company	varchar(max)	max
@RedTagType	char	1
@RedTagUse	varchar(500)	500

SQL Script

```
CREATE Proc [Report].[UspResults_AmendmentJnl]
    (
      @Company Varchar(Max)
    , @RedTagType Char(1)
    , @RedTagUse Varchar(500)
    )
As
    Begin
/*
Template designed by Chris Johnson, Prometic Group September 2015
///
Stored procedure set out to query multiple databases with the same information and
return it in a collated format        ///
*/
        If IsNumeric(@Company) = 0
            Begin
                Select  @Company = Upper(@Company);
            End;


        Set NoCount On;

--Red tag

        Declare @RedTagDB Varchar(255)= Db_Name();
        Exec [Process].[UspInsert_RedTagLogs] @StoredProcDb = 'BlackBox' ,
            @StoredProcSchema = 'Report' ,
            @StoredProcName = 'UspResults_AmendmentJnl' ,
            @UsedByType = @RedTagType , @UsedByName = @RedTagUse ,
            @UsedByDb = @RedTagDB;

--list the tables that are to be pulled back from each DB - if they are not found
the script will not be run against that db

        Declare @ListOfTables Varchar(Max) = 'ApAmendmentJnl';

--create temporary tables to be pulled from different databases, including a column
```

Author: Johnson, Chris

```
to id

        Create Table [#ApAmendmentJnl]
            (
              [Company] Varchar(50)             Collate Latin1_General_BIN
            , [JnlDate] DateTime2
            , [JnlTime] Decimal
            , [JnlLine] Decimal
            , [Supplier] Varchar(15)          Collate Latin1_General_BIN
            , [JournalPrinted] Char(1)          Collate Latin1_General_BIN
            , [ChangeFlag] Char(1)              Collate Latin1_General_BIN
            , [ColumnName] Varchar(50)          Collate Latin1_General_BIN
            , [Before] Varchar(255)              Collate Latin1_General_BIN
            , [After] Varchar(255)              Collate Latin1_General_BIN
            , [OperatorCode] Varchar(20)      Collate Latin1_General_BIN
            );

--create script to pull data from each db into the tables

        Declare @SQL Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end'
            + --Only query DBs beginning SysProCompany

            '
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN'
            + --only companies selected in main run, or if companies selected then
all

            '
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT'
            + --count number of tables requested (number of commas plus one)

            '
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')'
            + --Count of the tables requested how many exist in the db

            '
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'','')) '
            + --only if the count matches (all the tables exist in the requested db)
then run the script

            '
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
                    Insert #ApAmendmentJnl
                        ( Company
                        , JnlDate
```

```
                        , JnlTime
                        , JnlLine
                        , Supplier
                        , JournalPrinted
                        , ChangeFlag
                        , ColumnName
                        , Before
                        , After
                        , OperatorCode
                        )
                Select @DBCode
                    , JnlDate
                    , JnlTime
                    , JnlLine
                    , Supplier
                    , JournalPrinted
                    , ChangeFlag
                    , ColumnName
                    , Before
                    , After
                    , OperatorCode
                From
                ApAmendmentJnl With ( NoLock )
            End
    End';

--Enable this function to check script changes (try to run script directly against
db manually)

--Print @SQL


--execute script against each db, populating the base tables

        Exec [Process].[ExecForEachDB] @cmd = @SQL;

--define the results you want to return


--Placeholder to create indexes as required --*** not required as no joins are in
place***


--script to combine base data and insert into results table --*** Not required as is
direct dump***


--return results

        Select  [Company]
              , [JnlDate]
              , [JnlTime]
              , [JnlLine]
              , [Supplier]
              , [JournalPrinted]
              , [ChangeFlag]
              , [ColumnName]
              , [Before]
```

Author: Johnson, Chris

```
              , [After]
              , [OperatorCode]
        From    [#ApAmendmentJnl];

    End;


GO
EXEC sp_addextendedproperty N'MS_Description', N'details from the amendment jnl',
'SCHEMA', N'Report', 'PROCEDURE', N'UspResults_AmendmentJnl', NULL, NULL
GO
```

Uses

[Process].[ExecForEachDB]
[Process].[UspInsert_RedTagLogs]
[Report]

Author: Johnson, Chris

[Report].[UspResults_ApAgingInvoices]

MS_Description

replaced by UspResults_AccPayable_AgedInvoices

Parameters

Name	Data Type	Max Length (Bytes)
@RunDate	date	3
@RedTagType	char	1
@RedTagUse	varchar(500)	500

SQL Script

```
CREATE Proc [Report].[UspResults_ApAgingInvoices]
    (
      @RunDate Date
    , @RedTagType Char(1)
    , @RedTagUse Varchar(500)
    )
As
    Begin
/*
Template designed by Chris Johnson, Prometic Group March 2016
*/

--remove nocount on to speed up query

        Set NoCount On;

--Red tag

        Declare @RedTagDB Varchar(255)= Db_Name();
        Exec [Process].[UspInsert_RedTagLogs] @StoredProcDb = 'BlackBox' ,
            @StoredProcSchema = 'Report' ,
            @StoredProcName = 'UspResults_ApAgingInvoices' ,
            @UsedByType = @RedTagType , @UsedByName = @RedTagUse ,
            @UsedByDb = @RedTagDB;

--If no rundate defined, use todays date

        Select  @RunDate = Coalesce(@RunDate , GetDate());

--list the tables that are to be pulled back from each DB - if they are not found
the script will not be run against that db

        Declare @ListOfTables Varchar(Max) = 'AssetDepreciation,TblApTerms';

--create temporary tables to be pulled from different databases, including a column
to id
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_Ap-AgingInvoices

```
        Create Table [#ApInvoice]
            (
              [DB] Varchar(50) Collate Latin1_General_BIN
            , [Supplier] Varchar(50) Collate Latin1_General_BIN
            , [Invoice] Varchar(20) Collate Latin1_General_BIN
            , [NextPaymEntry] Int
            , [JournalDate] Date
            , [Journal] Int
            , [Branch] Varchar(10) Collate Latin1_General_BIN
            , [InvoiceDate] Date
            , [DiscountDate] Date
            , [DueDate] Date
            , [Reference] Varchar(30) Collate Latin1_General_BIN
            , [OrigInvValue] Numeric(20 , 2)
            , [OrigDiscValue] Numeric(20 , 2)
            , [MthInvBal1] Numeric(20 , 2)
            , [MthInvBal2] Numeric(20 , 2)
            , [MthInvBal3] Numeric(20 , 2)
            , [ManualChqDate] Date
            , [ManualChqNum] BigInt
            , [DiscActiveFlag] Char(1) Collate Latin1_General_BIN
            , [InvoiceStatus] Char(1) Collate Latin1_General_BIN
            , [Currency] Varchar(10) Collate Latin1_General_BIN
            , [Bank] Varchar(20) Collate Latin1_General_BIN
            , [PaymGrossValue] Numeric(20 , 2)
            , [PaymDiscValue] Numeric(20 , 2)
            , [TaxPortionDisc] Numeric(20 , 2)
            , [NotificationDate] Date
            , [InvoiceYear] Int
            , [InvoiceMonth] Int
            , [YearInvBalZero] Int
            , [MonthInvBalZero] Int
            , [ExchangeRate] Float
            , [FirstTaxCode] Char(3) Collate Latin1_General_BIN
            , [WithTaxValue] Numeric(20 , 2)
            , [WithTaxRate] Float
            , [CurrencyValue] Numeric(20 , 2)
            , [PostCurrency] Varchar(10) Collate Latin1_General_BIN
            , [ConvRate] Float
            , [MulDiv] Char(1) Collate Latin1_General_BIN
            , [AccountCur] Varchar(10) Collate Latin1_General_BIN
            , [AccConvRate] Float
            , [AccMulDiv] Char(1) Collate Latin1_General_BIN
            , [TriangCurrency] Varchar(10) Collate Latin1_General_BIN
            , [TriangRate] Float
            , [TriangMulDiv] Char(1) Collate Latin1_General_BIN
            , [Tax2Value] Numeric(20 , 2)
            , [VatInvalid] Char(1) Collate Latin1_General_BIN
            , [EntryNumber] Int
            , [PaymentChkType] Char(1) Collate Latin1_General_BIN
            , [ManualChRef] Varchar(30)
            , [PaymentNumber] Varchar(15) Collate Latin1_General_BIN
            , [InvoiceTakeOn] Char(1) Collate Latin1_General_BIN
            , [FixExchangeRate] Char(1) Collate Latin1_General_BIN
            , [NextRevalNo] Int
            , [TaxReverse] Char(1) Collate Latin1_General_BIN
            , [Tax2Reverse] Char(1) Collate Latin1_General_BIN
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_Ap-AgingInvoices

```
            , [NationalitySource] Char(3) Collate Latin1_General_BIN
            , [NationalityDest] Char(3) Collate Latin1_General_BIN
            , [AutoVoucherCreated] Char(1) Collate Latin1_General_BIN
            , [AutoVoucherPrinted] Char(1) Collate Latin1_General_BIN
            , [SecondTaxCode] Char(3) Collate Latin1_General_BIN
            , [WithTaxCode] Char(3) Collate Latin1_General_BIN
            );
        Create Table [#ApSupplier]
            (
              [DB] Varchar(50) Collate Latin1_General_BIN
            , [Supplier] Varchar(50) Collate Latin1_General_BIN
            , [SupplierName] Varchar(255) Collate Latin1_General_BIN
            , [TermsCode] Char(2) Collate Latin1_General_BIN
            );
        Create Table [#TblApTerms]
            (
              [DB] Varchar(50) Collate Latin1_General_BIN
            , [TermsCode] Char(2) Collate Latin1_General_BIN
            , [Description] Varchar(50) Collate Latin1_General_BIN
            );



--create script to pull data from each db into the tables

        Declare @SQLApInvoice Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end'
            + --Only query DBs beginning SysProCompany

            '
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN'
            + --only companies selected in main run, or if companies selected then
all

            '
        IF isnumeric(@DBCode)=1
            begin
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT'
            + --count number of tables requested (number of commas plus one)

            '
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')'
            + --Count of the tables requested how many exist in the db

            '
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'',''))  '
            + --only if the count matches (all the tables exist in the requested db)
then run the script
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_Ap-AgingInvoices

```
            '
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
            Insert [#ApInvoice]
        ( [DB], [Supplier], [Invoice], [NextPaymEntry], [JournalDate]
        , [Journal], [Branch], [InvoiceDate], [DiscountDate], [DueDate]
        , [Reference], [OrigInvValue], [OrigDiscValue], [MthInvBal1], [MthInvBal2]
        , [MthInvBal3], [ManualChqDate], [ManualChqNum], [DiscActiveFlag], [Invoice-
Status]
        , [Currency], [Bank], [PaymGrossValue], [PaymDiscValue], [TaxPortionDisc],
[NotificationDate]
        , [InvoiceYear], [InvoiceMonth], [YearInvBalZero], [MonthInvBalZero],
[ExchangeRate]
        , [FirstTaxCode], [WithTaxValue], [WithTaxRate], [CurrencyValue], [Post-
Currency], [ConvRate]
        , [MulDiv], [AccountCur], [AccConvRate], [AccMulDiv], [TriangCurrency],
[TriangRate]
        , [TriangMulDiv], [Tax2Value], [VatInvalid], [EntryNumber], [PaymentChk-
Type], [ManualChRef]
        , [PaymentNumber], [InvoiceTakeOn], [FixExchangeRate], [NextRevalNo], [Tax-
Reverse]
        , [Tax2Reverse], [NationalitySource], [NationalityDest], [AutoVoucher-
Created]
        , [AutoVoucherPrinted], [SecondTaxCode], [WithTaxCode]
        )
SELECT [DB]=@DBCode, [AI].[Supplier], [AI].[Invoice], [AI].[NextPaymEntry],
[AI].[JournalDate]
     , [AI].[Journal], [AI].[Branch], [AI].[InvoiceDate], [AI].[DiscountDate],
[AI].[DueDate]
     , [AI].[Reference], [AI].[OrigInvValue], [AI].[OrigDiscValue], [AI].[MthInv-
Bal1], [AI].[MthInvBal2]
     , [AI].[MthInvBal3], [AI].[ManualChqDate], [AI].[ManualChqNum], [AI].[Disc-
ActiveFlag], [AI].[InvoiceStatus]
     , [AI].[Currency], [AI].[Bank], [AI].[PaymGrossValue], [AI].[PaymDiscValue],
[AI].[TaxPortionDisc], [AI].[NotificationDate]
     , [AI].[InvoiceYear], [AI].[InvoiceMonth], [AI].[YearInvBalZero], [AI].[Month-
InvBalZero], [AI].[ExchangeRate]
     , [AI].[FirstTaxCode], [AI].[WithTaxValue], [AI].[WithTaxRate], [AI].[Currency-
Value], [AI].[PostCurrency], [AI].[ConvRate]
     , [AI].[MulDiv], [AI].[AccountCur], [AI].[AccConvRate], [AI].[AccMulDiv],
[AI].[TriangCurrency], [AI].[TriangRate]
     , [AI].[TriangMulDiv], [AI].[Tax2Value], [AI].[VatInvalid], [AI].[EntryNumber],
[AI].[PaymentChkType], [AI].[ManualChRef]
     , [AI].[PaymentNumber], [AI].[InvoiceTakeOn], [AI].[FixExchangeRate],
[AI].[NextRevalNo], [AI].[TaxReverse]
     , [AI].[Tax2Reverse], [AI].[NationalitySource], [AI].[NationalityDest],
[AI].[AutoVoucherCreated]
     , [AI].[AutoVoucherPrinted], [AI].[SecondTaxCode], [AI].[WithTaxCode]
     FROM [ApInvoice] As [AI]
     where [AI].[MthInvBal1]<>0
            End
        End
    End';
        Declare @SQLApSupplier Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end'
            + --Only query DBs beginning SysProCompany

            '
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
```

Author: Johnson, Chris

```
    BEGIN'
            + --only companies selected in main run, or if companies selected then
all

            '
        IF isnumeric(@DBCode)=1
            begin
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT'
            + --count number of tables requested (number of commas plus one)

            '
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')'
            + --Count of the tables requested how many exist in the db

            '
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'','')) '
            + --only if the count matches (all the tables exist in the requested db)
then run the script

            '
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
                        Insert [#ApSupplier]
                    ( [DB] , [Supplier] , [SupplierName], [TermsCode]  )
            SELECT [DB]=@DBCode
                 , [AS].[Supplier]
                 , [AS].[SupplierName]
                 , [TermsCode]  FROM [ApSupplier] As [AS]
            End
        End
    End';
        Declare @SQLTblApTerms Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end'
            + --Only query DBs beginning SysProCompany

            '
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN'
            + --only companies selected in main run, or if companies selected then
all

            '
        IF isnumeric(@DBCode)=1
            begin
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT'
            + --count number of tables requested (number of commas plus one)

            '
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
```

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_Ap-AgingInvoices

```
String](@ListOfTables,'','')'
            + --Count of the tables requested how many exist in the db

            '
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'','')) '
            + --only if the count matches (all the tables exist in the requested db)
then run the script

            '
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
                Insert [#TblApTerms]
                        ( [DB] , [TermsCode] , [Description] )
                SELECT [DB]=@DBCode
                     , [TAT].[TermsCode]
                     , [TAT].[Description] FROM [TblApTerms] As [TAT]
            End
        End
    End';
--Enable this function to check script changes (try to run script directly against
db manually)

--Print @SQL


--execute script against each db, populating the base tables

        Exec [Process].[ExecForEachDB] @cmd = @SQLApInvoice;
        Exec [Process].[ExecForEachDB] @cmd = @SQLApSupplier;
        Exec [Process].[ExecForEachDB] @cmd = @SQLTblApTerms;


--define the results you want to return

        Create Table [#ApData]
            (
              [Company] Varchar(50) Collate Latin1_General_BIN
            , [Supplier] Varchar(255) Collate Latin1_General_BIN
            , [Invoice] Varchar(20) Collate Latin1_General_BIN
            , [NextPaymEntry] Int
            , [JournalDate] Date
            , [Journal] Int
            , [Branch] Varchar(10) Collate Latin1_General_BIN
            , [InvoiceDate] Date
            , [DiscountDate] Date
            , [DueDate] Date
            , [Reference] Varchar(30) Collate Latin1_General_BIN
            , [OrigInvValue] Numeric(20 , 2)
            , [OrigDiscValue] Numeric(20 , 2)
            , [MthInvBal1] Numeric(20 , 2)
            , [MthInvBal2] Numeric(20 , 2)
            , [MthInvBal3] Numeric(20 , 2)
            , [ManualChqDate] Date
            , [ManualChqNum] BigInt
            , [DiscActiveFlag] Char(1) Collate Latin1_General_BIN
            , [InvoiceStatus] Char(1) Collate Latin1_General_BIN
```

Author: Johnson, Chris

```
            , [Currency] Varchar(10) Collate Latin1_General_BIN
            , [Bank] Varchar(20) Collate Latin1_General_BIN
            , [PaymGrossValue] Numeric(20 , 2)
            , [PaymDiscValue] Numeric(20 , 2)
            , [TaxPortionDisc] Numeric(20 , 2)
            , [NotificationDate] Date
            , [InvoiceYear] Int
            , [InvoiceMonth] Int
            , [YearInvBalZero] Int
            , [MonthInvBalZero] Int
            , [ExchangeRate] Float
            , [FirstTaxCode] Char(3) Collate Latin1_General_BIN
            , [WithTaxValue] Numeric(20 , 2)
            , [WithTaxRate] Float
            , [CurrencyValue] Numeric(20 , 2)
            , [PostCurrency] Varchar(10) Collate Latin1_General_BIN
            , [ConvRate] Float
            , [MulDiv] Char(1) Collate Latin1_General_BIN
            , [AccountCur] Varchar(10) Collate Latin1_General_BIN
            , [AccConvRate] Float
            , [AccMulDiv] Char(1) Collate Latin1_General_BIN
            , [TriangCurrency] Varchar(10) Collate Latin1_General_BIN
            , [TriangRate] Float
            , [TriangMulDiv] Char(1) Collate Latin1_General_BIN
            , [Tax2Value] Numeric(20 , 2)
            , [VatInvalid] Char(1) Collate Latin1_General_BIN
            , [EntryNumber] Int
            , [PaymentChkType] Char(1) Collate Latin1_General_BIN
            , [ManualChRef] Varchar(30) Collate Latin1_General_BIN
            , [PaymentNumber] Varchar(15) Collate Latin1_General_BIN
            , [InvoiceTakeOn] Char(1) Collate Latin1_General_BIN
            , [FixExchangeRate] Char(1) Collate Latin1_General_BIN
            , [NextRevalNo] Int
            , [TaxReverse] Char(1) Collate Latin1_General_BIN
            , [Tax2Reverse] Char(1) Collate Latin1_General_BIN
            , [NationalitySource] Char(3) Collate Latin1_General_BIN
            , [NationalityDest] Char(3) Collate Latin1_General_BIN
            , [AutoVoucherCreated] Char(1) Collate Latin1_General_BIN
            , [AutoVoucherPrinted] Char(1) Collate Latin1_General_BIN
            , [SecondTaxCode] Char(3) Collate Latin1_General_BIN
            , [WithTaxCode] Char(3) Collate Latin1_General_BIN
            , [30Days] Numeric(20 , 2)
            , [45Days] Numeric(20 , 2)
            , [60Days] Numeric(20 , 2)
            , [90Days] Numeric(20 , 2)
            , [120Days] Numeric(20 , 2)
            , [121DaysPlus] Numeric(20 , 2)
            , [LocalCurrency] Numeric(20 , 2)
            , [SupplierTerms] Varchar(50)
            , [RunDate] Date
            );

--Placeholder to create indexes as required


--script to combine base data and insert into results table
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_Ap-AgingInvoices

```
        Insert  [#ApData]
                ( [Company]
                , [Supplier]
                , [Invoice]
                , [NextPaymEntry]
                , [JournalDate]
                , [Journal]
                , [Branch]
                , [InvoiceDate]
                , [DiscountDate]
                , [DueDate]
                , [Reference]
                , [OrigInvValue]
                , [OrigDiscValue]
                , [MthInvBal1]
                , [MthInvBal2]
                , [MthInvBal3]
                , [ManualChqDate]
                , [ManualChqNum]
                , [DiscActiveFlag]
                , [InvoiceStatus]
                , [Currency]
                , [Bank]
                , [PaymGrossValue]
                , [PaymDiscValue]
                , [TaxPortionDisc]
                , [NotificationDate]
                , [InvoiceYear]
                , [InvoiceMonth]
                , [YearInvBalZero]
                , [MonthInvBalZero]
                , [ExchangeRate]
                , [FirstTaxCode]
                , [WithTaxValue]
                , [WithTaxRate]
                , [CurrencyValue]
                , [PostCurrency]
                , [ConvRate]
                , [MulDiv]
                , [AccountCur]
                , [AccConvRate]
                , [AccMulDiv]
                , [TriangCurrency]
                , [TriangRate]
                , [TriangMulDiv]
                , [Tax2Value]
                , [VatInvalid]
                , [EntryNumber]
                , [PaymentChkType]
                , [ManualChRef]
                , [PaymentNumber]
                , [InvoiceTakeOn]
                , [FixExchangeRate]
                , [NextRevalNo]
                , [TaxReverse]
                , [Tax2Reverse]
                , [NationalitySource]
```

Author: Johnson, Chris

```
                , [NationalityDest]
                , [AutoVoucherCreated]
                , [AutoVoucherPrinted]
                , [SecondTaxCode]
                , [WithTaxCode]
                , [30Days]
                , [45Days]
                , [60Days]
                , [90Days]
                , [120Days]
                , [121DaysPlus]
                , [LocalCurrency]
                , [SupplierTerms]
                , [RunDate]
                )
                Select  [Company] = [API].[DB]
                      , [Supplier] = LTrim(RTrim([API].[Supplier])) + ' - '
                        + LTrim(RTrim([APS].[SupplierName]))
                      , [API].[Invoice]
                      , [API].[NextPaymEntry]
                      , [API].[JournalDate]
                      , [API].[Journal]
                      , [API].[Branch]
                      , [API].[InvoiceDate]
                      , [API].[DiscountDate]
                      , [API].[DueDate]
                      , [API].[Reference]
                      , [API].[OrigInvValue]
                      , [API].[OrigDiscValue]
                      , [API].[MthInvBal1]
                      , [API].[MthInvBal2]
                      , [API].[MthInvBal3]
                      , [API].[ManualChqDate]
                      , [API].[ManualChqNum]
                      , [API].[DiscActiveFlag]
                      , [API].[InvoiceStatus]
                      , [API].[Currency]
                      , [API].[Bank]
                      , [API].[PaymGrossValue]
                      , [API].[PaymDiscValue]
                      , [API].[TaxPortionDisc]
                      , [API].[NotificationDate]
                      , [API].[InvoiceYear]
                      , [API].[InvoiceMonth]
                      , [API].[YearInvBalZero]
                      , [API].[MonthInvBalZero]
                      , [API].[ExchangeRate]
                      , [API].[FirstTaxCode]
                      , [API].[WithTaxValue]
                      , [API].[WithTaxRate]
                      , [API].[CurrencyValue]
                      , [API].[PostCurrency]
                      , [API].[ConvRate]
                      , [API].[MulDiv]
                      , [API].[AccountCur]
                      , [API].[AccConvRate]
                      , [API].[AccMulDiv]
```

Author: Johnson, Chris

```
                        , [API].[TriangCurrency]
                        , [API].[TriangRate]
                        , [API].[TriangMulDiv]
                        , [API].[Tax2Value]
                        , [API].[VatInvalid]
                        , [API].[EntryNumber]
                        , [API].[PaymentChkType]
                        , [API].[ManualChRef]
                        , [API].[PaymentNumber]
                        , [API].[InvoiceTakeOn]
                        , [API].[FixExchangeRate]
                        , [API].[NextRevalNo]
                        , [API].[TaxReverse]
                        , [API].[Tax2Reverse]
                        , [API].[NationalitySource]
                        , [API].[NationalityDest]
                        , [API].[AutoVoucherCreated]
                        , [API].[AutoVoucherPrinted]
                        , [API].[SecondTaxCode]
                        , [API].[WithTaxCode]
                        , [30Days] = Case When DateDiff(dd , [API].[InvoiceDate] ,
                                                        @RunDate) <= 30
                                          Then [API].[MthInvBal1]
                                          Else 0
                                     End
                        , [45Days] = Case When DateDiff(dd , [API].[InvoiceDate] ,
                                                        @RunDate) Between 31
                                                                          And
                                                                          45
                                          Then [API].[MthInvBal1]
                                          Else 0
                                     End
                        , [60Days] = Case When DateDiff(dd , [API].[InvoiceDate] ,
                                                        @RunDate) Between 46
                                                                          And
                                                                          60
                                          Then [API].[MthInvBal1]
                                          Else 0
                                     End
                        , [90Days] = Case When DateDiff(dd , [API].[InvoiceDate] ,
                                                        @RunDate) Between 61
                                                                          And
                                                                          90
                                          Then [API].[MthInvBal1]
                                          Else 0
                                     End
                        , [120Days] = Case When DateDiff(dd ,
                                                         [API].[InvoiceDate] ,
                                                         @RunDate) Between 91
                                                                           And
                                                                           120
                                           Then [API].[MthInvBal1]
                                           Else 0
                                      End
                        , [121DaysPlus] = Case When DateDiff(dd ,
                                                             [API].[InvoiceDate] ,
                                                             @RunDate) > 120
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_Ap-AgingInvoices

```
                                            Then [API].[MthInvBal1]
                                            Else 0
                                       End
                     , [LocalCurrency] = Cast(Case When [API].[MulDiv] = 'M'
                                                   Then [API].[MthInvBal1]
                                                        * [API].[ConvRate]
                                                   Else [API].[MthInvBal1]
                                                        / [API].[ConvRate]
                                              End As Decimal(20 , 2))
                     , [SupplierTerms] = [TAT].[Description]
                     , @RunDate
                From    [#ApInvoice] As [API]
                          Left Outer Join [#ApSupplier] As [APS] On [API].[Supplier] =
[APS].[Supplier]
                                                                    And [APS].[DB] =
[API].[DB]
                          Left Join [#TblApTerms] As [TAT] On [TAT].[TermsCode] =
[APS].[TermsCode]
                                                                  And [TAT].[DB] =
[APS].[DB];


--return results

        Select  [AD].[Company]
              , [AD].[Supplier]
              , [AD].[Invoice]
              , [AD].[NextPaymEntry]
              , [AD].[JournalDate]
              , [AD].[Journal]
              , [AD].[Branch]
              , [AD].[InvoiceDate]
              , [AD].[DiscountDate]
              , [AD].[DueDate]
              , [AD].[Reference]
              , [AD].[OrigInvValue]
              , [AD].[OrigDiscValue]
              , [AD].[MthInvBal1]
              , [AD].[MthInvBal2]
              , [AD].[MthInvBal3]
              , [AD].[ManualChqDate]
              , [AD].[ManualChqNum]
              , [AD].[DiscActiveFlag]
              , [AD].[InvoiceStatus]
              , [AD].[Currency]
              , [AD].[Bank]
              , [AD].[PaymGrossValue]
              , [AD].[PaymDiscValue]
              , [AD].[TaxPortionDisc]
              , [AD].[NotificationDate]
              , [AD].[InvoiceYear]
              , [AD].[InvoiceMonth]
              , [AD].[YearInvBalZero]
              , [AD].[MonthInvBalZero]
              , [AD].[ExchangeRate]
              , [AD].[FirstTaxCode]
              , [AD].[WithTaxValue]
              , [AD].[WithTaxRate]
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_Ap-AgingInvoices

```
                , [AD].[CurrencyValue]
                , [AD].[PostCurrency]
                , [AD].[ConvRate]
                , [AD].[MulDiv]
                , [AD].[AccountCur]
                , [AD].[AccConvRate]
                , [AD].[AccMulDiv]
                , [AD].[TriangCurrency]
                , [AD].[TriangRate]
                , [AD].[TriangMulDiv]
                , [AD].[Tax2Value]
                , [AD].[VatInvalid]
                , [AD].[EntryNumber]
                , [AD].[PaymentChkType]
                , [AD].[ManualChRef]
                , [AD].[PaymentNumber]
                , [AD].[InvoiceTakeOn]
                , [AD].[FixExchangeRate]
                , [AD].[NextRevalNo]
                , [AD].[TaxReverse]
                , [AD].[Tax2Reverse]
                , [AD].[NationalitySource]
                , [AD].[NationalityDest]
                , [AD].[AutoVoucherCreated]
                , [AD].[AutoVoucherPrinted]
                , [AD].[SecondTaxCode]
                , [AD].[WithTaxCode]
                , [AD].[30Days]
                , [AD].[45Days]
                , [AD].[60Days]
                , [AD].[90Days]
                , [AD].[120Days]
                , [AD].[121DaysPlus]
                , [AD].[LocalCurrency]
                , [CN].[CompanyName]
                , [AD].[SupplierTerms]
                , [AD].[RunDate]
        From    [#ApData] As [AD]
                  Left Join [Lookups].[CompanyNames] As [CN] On [CN].[Company] =
[AD].[Company];

    End;

GO
EXEC sp_addextendedproperty N'MS_Description', N'replaced by UspResults_AccPayable_-
AgedInvoices', 'SCHEMA', N'Report', 'PROCEDURE', N'UspResults_ApAgingInvoices',
NULL, NULL
GO
```

Uses

[Lookups].[CompanyNames]
[Process].[ExecForEachDB]
[Process].[UspInsert_RedTagLogs]
[Report]

Author: Johnson, Chris

[Report].[UspResults_APDaysToPayment]

MS_Description

returns details on how long it takes for AP to be settled

Parameters

Name	Data Type	Max Length (Bytes)
@Company	varchar(max)	max
@RedTagType	char	1
@RedTagUse	varchar(500)	500

SQL Script

```
CREATE Proc [Report].[UspResults_APDaysToPayment]
    (
      @Company Varchar(Max)
    , @RedTagType Char(1)
    , @RedTagUse Varchar(500)
    )
As
    Begin
/*
Template designed by Chris Johnson, Prometic Group September 2015
Stored procedure set out to query multiple databases with the same information and
return it in a collated format
*/
        If IsNumeric(@Company) = 0
            Begin
                Select  @Company = Upper(@Company);
            End;

--remove nocount on to speed up query

        Set NoCount On;

--Red tag

        Declare @RedTagDB Varchar(255)= Db_Name();
        Exec [Process].[UspInsert_RedTagLogs] @StoredProcDb = 'BlackBox' ,
            @StoredProcSchema = 'Report' ,
            @StoredProcName = 'UspResults_APDaysToPayment' ,
            @UsedByType = @RedTagType , @UsedByName = @RedTagUse ,
            @UsedByDb = @RedTagDB;

--list the tables that are to be pulled back from each DB - if they are not found
the script will not be run against that db

        Declare @ListOfTables Varchar(Max) = 'ApInvoice,ApInvoicePay,CshAp-
Payments,ApBank';
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-APDaysToPayment

```
--create temporary tables to be pulled from different databases, including a column
to id

        Create Table [#ApInvoice]
            (
              [DatabaseName] Varchar(150)        collate latin1_general_bin
            , [Supplier] Varchar(15)            collate latin1_general_bin
            , [Invoice] Varchar(20)               collate latin1_general_bin
            , [InvoiceDate] Date
            , [CurrencyValue] Numeric(20 , 2)
            , [ConvRate] Numeric(20 , 6)
            , [MulDiv] Char(1)                    collate latin1_general_bin
            , [PostCurrency] Varchar(3)           collate latin1_general_bin
            , [DueDate] Date
            );
        Create Table [#ApInvoicePay]
            (
              [DatabaseName] Varchar(150)        collate latin1_general_bin
            , [Supplier] Varchar(15)            collate latin1_general_bin
            , [JournalDate] Date
            , [TrnValue] Numeric(20 , 2)
            , [PaymentReference] Varchar(50)    collate latin1_general_bin
            , [PaymentNumber] Varchar(15)       collate latin1_general_bin
            , [PostCurrency] Varchar(3)           collate latin1_general_bin
            , [PostConvRate] Numeric(20 , 6)
            , [PostMulDiv] Char(1)                collate latin1_general_bin
            , [Journal] Int
            , [TrnType] Char(1)                   collate latin1_general_bin
            , [Invoice] Varchar(20)               collate latin1_general_bin
            );
        Create Table [#CshApPayments]
            (
              [DatabaseName] Varchar(150)        collate latin1_general_bin
            , [CbTrnDate] Date
            , [GrossValue] Numeric(20 , 2)
            , [NetValue] Numeric(20 , 2)
            , [Supplier] Varchar(15)            collate latin1_general_bin
            , [Invoice] Varchar(20)               collate latin1_general_bin
            , [PaymentNumber] Varchar(15)       collate latin1_general_bin
            , [Bank] Varchar(15)                collate latin1_general_bin
            );
        Create Table [#ApBank]
            (
              [DatabaseName] Varchar(150)        collate latin1_general_bin
            , [Description] Varchar(50)           collate latin1_general_bin
            , [Bank] Varchar(15)                collate latin1_general_bin
            );



--create script to pull data from each db into the tables

        Declare @SQL Varchar(Max) = 'USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-APDaysToPayment

```
    BEGIN
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''+
Upper(@Company)+ ''' = ''ALL''
            BEGIN
                    Insert [#ApBank]
                ( [DatabaseName], [Description], [Bank])
                    SELECT [DatabaseName]=@DBCode
                         , [AB].[Description]
                         , [AB].[Bank] FROM [ApBank] [AB]

                    Insert [#ApInvoice]
                ( [DatabaseName], [Supplier], [Invoice], [InvoiceDate], [Currency-
Value], [ConvRate], [MulDiv], [PostCurrency], [DueDate])
                    SELECT [DatabaseName]=@DBCode
                         , [AI].[Supplier]
                         , [AI].[Invoice]
                         , [AI].[InvoiceDate]
                         , [AI].[CurrencyValue]
                         , [AI].[ConvRate]
                         , [AI].[MulDiv]
                         , [AI].[PostCurrency]
                         , [AI].[DueDate] FROM [ApInvoice] [AI]

                    Insert [#ApInvoicePay]
                ( [DatabaseName], [Supplier], [JournalDate], [TrnValue], [Payment-
Reference], [PaymentNumber], [PostCurrency], [PostConvRate], [PostMulDiv],
[Journal], [TrnType], [Invoice])
                    SELECT [DatabaseName]=@DBCode
                         , [AIP].[Supplier]
                         , [AIP].[JournalDate]
                         , [AIP].[TrnValue]
                         , [AIP].[PaymentReference]
                         , [AIP].[PaymentNumber]
                         , [AIP].[PostCurrency]
                         , [AIP].[PostConvRate]
                         , [AIP].[PostMulDiv]
                         , [AIP].[Journal]
                         , [AIP].[TrnType]
                         , [AIP].[Invoice] FROM [ApInvoicePay] [AIP]

                    Insert [#CshApPayments]
                ( [DatabaseName], [CbTrnDate], [GrossValue], [NetValue], [Supplier],
[Invoice], [PaymentNumber], [Bank])
                    SELECT [DatabaseName]=@DBCode
                         , [CAP].[CbTrnDate]
                         , [CAP].[GrossValue]
                         , [CAP].[NetValue]
                         , [CAP].[Supplier]
                         , [CAP].[Invoice]
                         , [CAP].[PaymentNumber]
                         , [CAP].[Bank] FROM [CshApPayments] [CAP]
            End
    End';

--execute script against each db, populating the base tables

        Exec [Process].[ExecForEachDB_WithTableCheck] @cmd = @SQL , -- nvarchar(max)
```

Author: Johnson, Chris

```
            @SchemaTablesToCheck = @ListOfTables;

--define the results you want to return

        Create Table [#Results]
            (
              [Supplier] Varchar(15)                collate latin1_general_bin
            --, [SalesOrder] Varchar(20)             collate latin1_general_bin

            , [Invoice] Varchar(20)                   collate latin1_general_bin
            , [InvoiceDate] Date
            , [DueDate] Date
            , [CurrencyValue] Numeric(20 , 2)
            , [ConvRate] Numeric(20 , 6)
            , [MulDiv] Varchar(1)                     collate latin1_general_bin
            , [PostCurrency] Varchar(3)               collate latin1_general_bin
            , [LocalValue] Numeric(20 , 2)
            , [JournalDate] Date
            , [TrnTypeDesc] Varchar(250)              collate latin1_general_bin
            , [TrnValue] Numeric(20 , 2)
            , [InvoiceNotation] Varchar(50)           collate latin1_general_bin
            , [JournalPostCurrency] Varchar(3)        collate latin1_general_bin
            , [PostConvRate] Numeric(20 , 6)
            , [PostMulDiv] Varchar(1)                 collate latin1_general_bin
            , [LocalJournalValue] Numeric(20 , 2)
            , [Journal] Int
            , [CbTrnDate] Date
            , [GrossPayment] Numeric(20 , 2)
            , [NetPayment] Numeric(20 , 2)
            , [Bank] Varchar(50)                      collate latin1_general_bin
            , [Company] Varchar(150)                  collate latin1_general_bin
            , [CompanyName] Varchar(250)              collate latin1_general_bin
            );

        Insert  [#Results]
                ( [Supplier]
                , [Invoice]
                , [InvoiceDate]
                , [DueDate]
                , [CurrencyValue]
                , [ConvRate]
                , [MulDiv]
                , [PostCurrency]
                , [LocalValue]
                , [JournalDate]
                , [TrnTypeDesc]
                , [TrnValue]
                , [InvoiceNotation]
                , [JournalPostCurrency]
                , [PostConvRate]
                , [PostMulDiv]
                , [LocalJournalValue]
                , [Journal]
                , [CbTrnDate]
                , [GrossPayment]
                , [NetPayment]
                , [Bank]
```

Author: Johnson, Chris

```
                , [Company]
                , [CompanyName]
                )
                Select  [AI].[Supplier]
                      --, [AI].[SalesOrder]

                      , [AI].[Invoice]
                      , [AI].[InvoiceDate]
                      , [AI].[DueDate]
                      , [AI].[CurrencyValue]
                      , [AI].[ConvRate]
                      , [AI].[MulDiv]
                      , [AI].[PostCurrency]
                      , [LocalValue] = Case When [AI].[MulDiv] = 'D'
                                            Then Convert(Numeric(20 , 2) ,
[AI].[CurrencyValue]
                                                 / [AI].[ConvRate])
                                            Else Convert(Numeric(20 , 2) ,
[AI].[CurrencyValue]
                                                 * [AI].[ConvRate])
                                       End
                      , [JournalDate] = Convert(Date , [AIP].[JournalDate])
                      , [AIPTT].[TrnTypeDesc]
                      , [AIP].[TrnValue]
                      , [AIP].[PaymentReference]
                      , [JournalPostCurrency] = [AIP].[PostCurrency]
                      , [AIP].[PostConvRate]
                      , [AIP].[PostMulDiv]
                      , [LocalJournalValue] = Case When [AIP].[PostMulDiv] = 'D'
                                                   Then Convert(Numeric(20 , 2) ,
[AIP].[TrnValue]
                                                        / [AIP].[PostConvRate])
                                                   Else Convert(Numeric(20 , 2) ,
[AIP].[TrnValue]
                                                        * [AIP].[PostConvRate])
                                              End
                      , [AIP].[Journal]
                      , [CAP].[CbTrnDate]
                      , [CAP].[GrossValue]
                      , [CAP].[NetValue]
                      , [Bank] = [AB].[Description]
                      , [CN].[Company]
                      , [CN].[CompanyName]
                From    [#ApInvoice] [AI]
                        Left Join [#ApInvoicePay] [AIP]
                            On [AIP].[Supplier] = [AI].[Supplier]
                               And [AIP].[Invoice] = [AI].[Invoice]
                               And [AIP].[DatabaseName] = [AI].[DatabaseName]
                        Left Join [BlackBox].[Lookups].[ArInvoicePayTrnType] [AIPTT]
                            On [AIPTT].[TrnType] = [AIP].[TrnType]
                        Left Join [#CshApPayments] [CAP]
                            On [CAP].[Supplier] = [AIP].[Supplier]
                               And [CAP].[Invoice] = [AIP].[Invoice]
                               And [CAP].[PaymentNumber] = [AIP].[PaymentNumber]
                               And [CAP].[DatabaseName] = [AIP].[DatabaseName]
                        Left Join [#ApBank] [AB]
                            On [AB].[Bank] = [CAP].[Bank]
                               And [AB].[DatabaseName] = [CAP].[DatabaseName]
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-APDaysToPayment

```
                        Left Join [BlackBox].[Lookups].[CompanyNames] [CN]
                            On [CN].[Company] = [AI].[DatabaseName];

        Set NoCount Off;

        Select  [R].[InvoiceDate]
              , [R].[DueDate]
              , [R].[Invoice]
              , [R].[Supplier]
              , [R].[LocalValue]
              , [R].[CurrencyValue]
              , [ReceivedValue] = Sum([R].[LocalJournalValue])
              , [LatestDate] = Max([R].[JournalDate])
              , [ClosedDate] = Case When Abs([R].[LocalValue]) <> Abs(Sum([R].[Local-
JournalValue]))
                                    Then Null
                                    Else Max([R].[JournalDate])
                               End
              , [DaysToCloseFromInvoiceDate] = DateDiff(Day ,
                                                        [R].[InvoiceDate] ,
                                                        Max([R].[JournalDate]))
              , [DaysToCloseFromDueDate] = DateDiff(Day , [R].[DueDate] ,
                                                    Max([R].[JournalDate]))
              , [R].[CompanyName]
        From    [#Results] [R]
        Group By [R].[InvoiceDate]
              , [R].[DueDate]
              , [R].[Invoice]
              , [R].[Supplier]
              , [R].[LocalValue]
              , [R].[CurrencyValue]
              , [R].[CompanyName]
        Order By [R].[InvoiceDate] Desc;

    End;
GO
EXEC sp_addextendedproperty N'MS_Description', N'returns details on how long it
takes for AP to be settled', 'SCHEMA', N'Report', 'PROCEDURE', N'UspResults_APDaysTo-
Payment', NULL, NULL
GO
```

Uses

[Lookups].[ArInvoicePayTrnType]
[Lookups].[CompanyNames]
[Process].[ExecForEachDB_WithTableCheck]
[Process].[UspInsert_RedTagLogs]
[Report]

Author: Johnson, Chris

[Report].[UspResults_ApGlDisburse]

MS_Description

details of the general ledger distribution from accounts payable

Parameters

Name	Data Type	Max Length (Bytes)
@Company	varchar(max)	max
@RedTagType	char	1
@RedTagUse	varchar(500)	500

SQL Script

```
CREATE Proc [Report].[UspResults_ApGlDisburse]
    (
      @Company Varchar(Max)
    , @RedTagType Char(1)
    , @RedTagUse Varchar(500)
    )
As
    Begin
        Set NoCount On;

        If IsNumeric(@Company) = 0
            Begin
                Select  @Company = Upper(@Company);
            End;

        Declare @CompanyCommas Varchar(Max) = Replace(@Company , ',' , ''',''');


--Red tag

        Declare @RedTagDB Varchar(255)= Db_Name();
        Exec [Process].[UspInsert_RedTagLogs] @StoredProcDb = 'BlackBox' ,
            @StoredProcSchema = 'Report' ,
            @StoredProcName = 'UspResults_Template' ,
            @UsedByType = @RedTagType , @UsedByName = @RedTagUse ,
            @UsedByDb = @RedTagDB;

--list the tables that are to be pulled back from each DB - if they are not found
the script will not be run against that db

        Declare @ListOfTables Varchar(Max) = 'ApGlDisburse';

--create temporary tables to be pulled from different databases, including a column
to id

        Create Table [#ApGlDisburse]
            (
```

Author: Johnson, Chris

```
              [DatabaseName] Varchar(150) Collate Latin1_General_BIN
            , [DistrEntry] Int
            , [GlCode] Varchar(35) Collate Latin1_General_BIN
            , [GlIntPeriod] Int
            , [GlIntYear] Int
            , [GlJournal] BigInt
            , [GlPeriod] Int
            , [GlYear] Int
            , [PostConvRate] Float
            , [PostCurrency] Varchar(10) Collate Latin1_General_BIN
            , [PostMulDiv] Varchar(10) Collate Latin1_General_BIN
            , [SupplierName] Varchar(50) Collate Latin1_General_BIN
            , [Supplier] Varchar(15) Collate Latin1_General_BIN
            , [TriangConvRate] Float
            , [TriangCurrency] Varchar(10) Collate Latin1_General_BIN
            , [TriangMulDiv] Varchar(5) Collate Latin1_General_BIN
            , [TrnValue] Numeric(20 , 3)
            , [Invoice] Varchar(20) Collate Latin1_General_BIN
            );




--create script to pull data from each db into the tables

        Declare @SQL Varchar(Max) = 'USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN
        IF @DBCode in (''' + @CompanyCommas + ''') or ''' + @Company
            + ''' = ''ALL''
            BEGIN
                Insert [#ApGlDisburse]
                        ( [DatabaseName]
                        , [DistrEntry]
                        , [GlCode]
                        , [GlIntPeriod]
                        , [GlIntYear]
                        , [GlJournal]
                        , [GlPeriod]
                        , [GlYear]
                        , [PostConvRate]
                        , [PostCurrency]
                        , [PostMulDiv]
                        , [SupplierName]
                        , [Supplier]
                        , [TriangConvRate]
                        , [TriangCurrency]
                        , [TriangMulDiv]
                        , [TrnValue]
                        , [Invoice]
                        )
                SELECT [DatabaseName]=@DBCode
                     , [AGD].[DistrEntry]
                     , [AGD].[GlCode]
```

Author: Johnson, Chris

```
                    , [AGD].[GlIntPeriod]
                    , [AGD].[GlIntYear]
                    , [AGD].[GlJournal]
                    , [AGD].[GlPeriod]
                    , [AGD].[GlYear]
                    , [AGD].[PostConvRate]
                    , [AGD].[PostCurrency]
                    , [AGD].[PostMulDiv]
                    , [AGD].[SupplierName]
                    , [AGD].[Supplier]
                    , [AGD].[TriangConvRate]
                    , [AGD].[TriangCurrency]
                    , [AGD].[TriangMulDiv]
                    , [AGD].[TrnValue]
                    , [AGD].[Invoice]
                FROM [ApGlDisburse] [AGD]
            End
    End';

--Enable this function to check script changes (try to run script directly against
db manually)

--Print @SQL


--execute script against each db, populating the base tables

        Exec [Process].[ExecForEachDB_WithTableCheck] @cmd = @SQL ,
            @SchemaTablesToCheck = @ListOfTables;


--define the results you want to return



--Placeholder to create indexes as required


--script to combine base data and insert into results table



        Set NoCount Off;
--return results

        Select  [AGD].[DatabaseName]
              , [AGD].[DistrEntry]
              , [AGD].[GlCode]
              , [AGD].[GlIntPeriod]
              , [AGD].[GlIntYear]
              , [AGD].[GlJournal]
              , [AGD].[GlPeriod]
              , [AGD].[GlYear]
              , [AGD].[PostConvRate]
              , [AGD].[PostCurrency]
              , [AGD].[PostMulDiv]
              , [AGD].[SupplierName]
```

```
                , [AGD].[Supplier]
                , [AGD].[TriangConvRate]
                , [AGD].[TriangCurrency]
                , [AGD].[TriangMulDiv]
                , [AGD].[TrnValue]
                , [AGD].[Invoice]
        From    [#ApGlDisburse] [AGD];

    End;


GO
EXEC sp_addextendedproperty N'MS_Description', N'details of the general ledger
distribution from accounts payable', 'SCHEMA', N'Report', 'PROCEDURE', N'UspResults_-
ApGlDisburse', NULL, NULL
GO
```

Uses

[Process].[ExecForEachDB_WithTableCheck]
[Process].[UspInsert_RedTagLogs]
[Report]

Author: Johnson, Chris

[Report].[UspResults_ApInvoice]

MS_Description

details of Ap Invoices

Parameters

Name	Data Type	Max Length (Bytes)
@Company	varchar(max)	max
@RedTagType	char	1
@RedTagUse	varchar(500)	500

SQL Script

```
CREATE Proc [Report].[UspResults_ApInvoice]
    (
      @Company Varchar(Max)
    , @RedTagType Char(1)
    , @RedTagUse Varchar(500)
    )
As
    Begin
/*
Template designed by Chris Johnson, Prometic Group September 2015
Stored procedure set out to query multiple databases with the same information and
return it in a collated format
*/
        Set NoCount On;

        If IsNumeric(@Company) = 0
            Begin
                Select  @Company = Upper(@Company);
            End;

--Red tag

        Declare @RedTagDB Varchar(255)= Db_Name();
        Exec [Process].[UspInsert_RedTagLogs] @StoredProcDb = 'BlackBox' ,
            @StoredProcSchema = 'Report' ,
            @StoredProcName = 'UspResults_ApInvoice' ,
            @UsedByType = @RedTagType , @UsedByName = @RedTagUse ,
            @UsedByDb = @RedTagDB;

--list the tables that are to be pulled back from each DB - if they are not found
the script will not be run against that db

        Declare @ListOfTables Varchar(Max) = 'ApInvoice';

--create temporary tables to be pulled from different databases, including a column
to id
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_Ap-Invoice

```
        Create Table [#ApInvoice]
            (
              [DatabaseName] Varchar(150) Collate Latin1_General_BIN
            , [Supplier] Varchar(15) Collate Latin1_General_BIN
            , [Invoice] Varchar(20) Collate Latin1_General_BIN
            , [JournalDate] Date
            , [Journal] Int
            , [InvoiceDate] Date
            , [DiscountDate] Date
            , [DueDate] Date
            , [Reference] Varchar(30) Collate Latin1_General_BIN
            , [InvoiceStatus] Char(1) Collate Latin1_General_BIN
            , [Currency] Char(3) Collate Latin1_General_BIN
            , [InvoiceYear] Int
            , [InvoiceMonth] Int
            , [ExchangeRate] Numeric(15 , 8)
            , [CurrencyValue] Numeric(20 , 2)
            , [PostCurrency] Char(3) Collate Latin1_General_BIN
            , [ConvRate] Numeric(15 , 8)
            , [PaymentNumber] Varchar(15) Collate Latin1_General_BIN
            );



--create script to pull data from each db into the tables

        Declare @SQL Varchar(Max) = 'USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            BEGIN
                Insert [#ApInvoice]
                        ( [DatabaseName]
                        , [Supplier]
                        , [Invoice]
                        , [JournalDate]
                        , [Journal]
                        , [InvoiceDate]
                        , [DiscountDate]
                        , [DueDate]
                        , [Reference]
                        , [InvoiceStatus]
                        , [Currency]
                        , [InvoiceYear]
                        , [InvoiceMonth]
                        , [ExchangeRate]
                        , [CurrencyValue]
                        , [PostCurrency]
                        , [ConvRate]
                        , [PaymentNumber]
                        )
                SELECT @DBCode
                        , [AI].[Supplier]
```

Author: Johnson, Chris

```
                        , [AI].[Invoice]
                        , [AI].[JournalDate]
                        , [AI].[Journal]
                        , [AI].[InvoiceDate]
                        , [AI].[DiscountDate]
                        , [AI].[DueDate]
                        , [AI].[Reference]
                        , [AI].[InvoiceStatus]
                        , [AI].[Currency]
                        , [AI].[InvoiceYear]
                        , [AI].[InvoiceMonth]
                        , [AI].[ExchangeRate]
                        , [AI].[CurrencyValue]
                        , [AI].[PostCurrency]
                        , [AI].[ConvRate]
                        , [AI].[PaymentNumber]
                FROM [dbo].[ApInvoice] [AI]
            End
    End';

--Enable this function to check script changes (try to run script directly against
db manually)

--Print @SQL


--execute script against each db, populating the base tables

        Exec [Process].[ExecForEachDB_WithTableCheck] @cmd = @SQL ,
            @SchemaTablesToCheck = @ListOfTables;


--define the results you want to return


--Placeholder to create indexes as required


--script to combine base data and insert into results table



        Set NoCount Off;
--return results

        Select  [AI].[DatabaseName]
              , [AI].[Supplier]
              , [AI].[Invoice]
              , [AI].[JournalDate]
              , [AI].[Journal]
              , [AI].[InvoiceDate]
              , [AI].[DiscountDate]
              , [AI].[DueDate]
              , [AI].[Reference]
              , [AI].[InvoiceStatus]
              , [AI].[Currency]
              , [AI].[InvoiceYear]
```

```
                , [AI].[InvoiceMonth]
                , [AI].[ExchangeRate]
                , [AI].[CurrencyValue]
                , [AI].[PostCurrency]
                , [AI].[ConvRate]
                , [AI].[PaymentNumber]
        From    [#ApInvoice] [AI];

    End;

GO
EXEC sp_addextendedproperty N'MS_Description', N'details of Ap Invoices', 'SCHEMA',
N'Report', 'PROCEDURE', N'UspResults_ApInvoice', NULL, NULL
GO
```

Uses

[Process].[ExecForEachDB_WithTableCheck]
[Process].[UspInsert_RedTagLogs]
[Report]

Author: Johnson, Chris

[Report].[UspResults_ApInvoicesWithPaymentDetails]

MS_Description

details of Ap Invoices including payment details

Parameters

Name	Data Type	Max Length (Bytes)
@Company	varchar(max)	max
@RedTagType	char	1
@RedTagUse	varchar(500)	500

SQL Script

```
CREATE Proc [Report].[UspResults_ApInvoicesWithPaymentDetails]
    (
      @Company Varchar(Max)
    , @RedTagType Char(1)
    , @RedTagUse Varchar(500)
    )
As
    Begin
/*
Template designed by Chris Johnson, Prometic Group September 2015
Stored procedure set out to query multiple databases with the same information and
return it in a collated format
--exec [Report].[UspResults_ApInvoicesWithPaymentDetails] @Company =10
*/
        Set NoCount On;
        If IsNumeric(@Company) = 0
            Begin
                Select  @Company = Upper(@Company);
            End;

        If @Company Like '%,%'
            Begin
                Select  @Company = Replace(@Company , ',' , ''',''');
            End;

--Red tag

        Declare @RedTagDB Varchar(255)= Db_Name();
        Exec [Process].[UspInsert_RedTagLogs] @StoredProcDb = 'BlackBox' ,
            @StoredProcSchema = 'Report' ,
            @StoredProcName = 'UspResults_ApInvoicesWithPaymentDetails' ,
            @UsedByType = @RedTagType , @UsedByName = @RedTagUse ,
            @UsedByDb = @RedTagDB;

--list the tables that are to be pulled back from each DB - if they are not found
the script will not be run against that db
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_Ap-InvoicesWithPaymentDetails

```
        Declare @ListOfTables Varchar(Max) = 'ApInvoice,ApInvoicePay,ApSupplier';
--create temporary tables to be pulled from different databases, including a column to id

        Create Table [#ApInvoice]
            (
              [DatabaseName] Varchar(150)        collate latin1_general_bin
            , [Supplier] Varchar(35)            collate latin1_general_bin
            , [Invoice] Varchar(150)            collate latin1_general_bin
            , [InvoiceDate] DateTime2
            , [DueDate] DateTime2
            , [OrigInvValue] Numeric(20 , 7)
            , [OrigDiscValue] Numeric(20 , 7)
            , [PaymentNumber] Varchar(150)        collate latin1_general_bin
            , [MthInvBal1] Numeric(20 , 7)
            , [MthInvBal2] Numeric(20 , 7)
            , [MthInvBal3] Numeric(20 , 7)
            , [ConvRate] Numeric(20 , 7)
            , [Currency] Char(3)                collate latin1_general_bin
            , [Journal] BigInt
            );
        Create Table [#ApInvoicePay]
            (
              [DatabaseName] Varchar(150)        collate latin1_general_bin
            , [Supplier] Varchar(35)            collate latin1_general_bin
            , [Invoice] Varchar(150)            collate latin1_general_bin
            , [PaymentReference] Varchar(150)    collate latin1_general_bin
            , [PostValue] Numeric(20 , 7)
            );
        Create Table [#ApSupplier]
            (
              [DatabaseName] Varchar(150)        collate latin1_general_bin
            , [Supplier] Varchar(50)            collate latin1_general_bin
            , [SupplierName] Varchar(150)        collate latin1_general_bin
            );

--create script to pull data from each db into the tables

        Declare @SQL1 Varchar(Max) = 'USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-Name(),len(db_Name())-13) else null end
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN
        IF @DBCode in (''' + @Company + ''') or '''
            + Replace(QuoteName(@Company) , '''' , '') + ''' = ''[ALL]''
            BEGIN
                Insert [#ApInvoice]
                        ( [DatabaseName]
                        , [Supplier]
                        , [Invoice]
                        , [InvoiceDate]
                        , [DueDate]
                        , [OrigInvValue]
                        , [OrigDiscValue]
                        , [PaymentNumber]
                        , [MthInvBal1]
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_Ap-InvoicesWithPaymentDetails

```
                        , [MthInvBal2]
                        , [MthInvBal3]
                        , [ConvRate]
                        , [Currency]
                        , [Journal]
                        )
                SELECT [DatabaseName]=@DBCode
                     , [ai].[Supplier]
                     , [ai].[Invoice]
                     , [ai].[InvoiceDate]
                     , [ai].[DueDate]
                     , [ai].[OrigInvValue]
                     , [ai].[OrigDiscValue]
                     , [ai].[PaymentNumber]
                     , [ai].[MthInvBal1]
                     , [ai].[MthInvBal2]
                     , [ai].[MthInvBal3]
                     , [ai].[ConvRate]
                     , [ai].[Currency]
                     , [ai].[Journal]
                From [ApInvoice] As [ai]
            End
    End';
        Declare @SQL2 Varchar(Max) = 'USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN
        IF @DBCode in (''' + @Company + ''') or '''
            + Replace(QuoteName(@Company) , '''' , '') + ''' = ''[ALL]''
            BEGIN
                Insert [#ApInvoicePay]
                        ( [DatabaseName]
                        , [Supplier]
                        , [Invoice]
                        , [PaymentReference]
                        , [PostValue]
                        )
                SELECT [DatabaseName]=@DBCode
                     , [aip].[Supplier]
                     , [aip].[Invoice]
                     , [aip].[PaymentReference]
                     , [aip].[PostValue]
                From [ApInvoicePay] As [aip]
            End
    End';
        Declare @SQL3 Varchar(Max) = 'USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN
        IF @DBCode in (''' + @Company + ''') or '''
            + Replace(QuoteName(@Company) , '''' , '') + ''' = ''[ALL]''
            BEGIN
                Insert [#ApSupplier]
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_Ap-InvoicesWithPaymentDetails

```
                        ( [DatabaseName]
                        , [Supplier]
                        , [SupplierName]
                        )
                SELECT [DatabaseName]=@DBCode
                     , [as].[Supplier]
                     , [as].[SupplierName]
                FROM [ApSupplier] As [as]
            End
    End';
--Enable this function to check script changes (try to run script directly against
db manually)

--Print @SQL


--execute script against each db, populating the base tables

        --Print 1

        Exec [Process].[ExecForEachDB_WithTableCheck] @cmd = @SQL1 , --
nvarchar(max)

            @SchemaTablesToCheck = @ListOfTables; -- nvarchar(max)

        --Print 2

        Exec [Process].[ExecForEachDB_WithTableCheck] @cmd = @SQL2 , --
nvarchar(max)

            @SchemaTablesToCheck = @ListOfTables; -- nvarchar(max)

        --Print 3

        Exec [Process].[ExecForEachDB_WithTableCheck] @cmd = @SQL3 , --
nvarchar(max)

            @SchemaTablesToCheck = @ListOfTables; -- nvarchar(max)

        --Print 4

--define the results you want to return

        Create Table [#Results]
            (
              [DatabaseName] Varchar(150)        collate latin1_general_bin
            , [Supplier] Varchar(30)            collate latin1_general_bin
            , [SupplierName] Varchar(150)        collate latin1_general_bin
            , [Invoice] Varchar(50)                collate latin1_general_bin
            , [InvoiceDate] DateTime2
            , [DueDate] DateTime2
            , [OrigInvValue] Numeric(20 , 7)
            , [OrigDiscValue] Numeric(20 , 7)
            , [PaymentReference] Varchar(150)    collate latin1_general_bin
            , [PaymentNumber] Varchar(150)        collate latin1_general_bin
            , [PostValue] Numeric(20 , 7)
            , [Value] Numeric(20 , 7)
            , [MthInvBal1] Numeric(20 , 7)
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_Ap-InvoicesWithPaymentDetails

```
            , [MthInvBal2] Numeric(20 , 7)
            , [MthInvBal3] Numeric(20 , 7)
            , [ConvRate] Numeric(20 , 7)
            , [Currency] Char(3)                collate latin1_general_bin
            , [Journal] BigInt
            );
        Create Table [#SupplierSummary]
            (
              [DatabaseName] Varchar(150)        Collate Latin1_General_BIN
            , [Supplier] Varchar(50)            collate latin1_general_bin
            , [SupplierValue] Numeric(20 , 7)
            , [Currency] Char(3)                collate latin1_general_bin
            );


--Placeholder to create indexes as required


--script to combine base data and insert into results table


        --Print 5

        Insert  [#Results]
                ( [DatabaseName]
                , [Supplier]
                , [Invoice]
                , [InvoiceDate]
                , [DueDate]
                , [OrigInvValue]
                , [OrigDiscValue]
                , [PaymentReference]
                , [PaymentNumber]
                , [PostValue]
                , [Value]
                , [MthInvBal1]
                , [MthInvBal2]
                , [MthInvBal3]
                , [SupplierName]
                , [ConvRate]
                , [Currency]
                , [Journal]
                )
                Select  [ai].[DatabaseName]
                      , [ai].[Supplier]
                      , [ai].[Invoice]
                      , [ai].[InvoiceDate]
                      , [ai].[DueDate]
                      , [ai].[OrigInvValue]
                      , [ai].[OrigDiscValue]
                      , [PaymentReference] = Min([aip].[PaymentReference])
                      , [PaymentNumber] = Max([ai].[PaymentNumber])
                      , [PostValue] = Sum([aip].[PostValue])
                      , [Value] = [ai].[OrigInvValue] + Sum([aip].[PostValue])
                      , [ai].[MthInvBal1]
                      , [ai].[MthInvBal2]
                      , [ai].[MthInvBal3]
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_Ap-InvoicesWithPaymentDetails

```
                      , [as].[SupplierName]
                      , [ai].[ConvRate]
                      , [ai].[Currency]
                      , [ai].[Journal]
                From    [#ApInvoice] As [ai]
                        Left Join [#ApInvoicePay] As [aip]
                            On [aip].[Supplier] = [ai].[Supplier]
                               And [aip].[Invoice] = [ai].[Invoice]
                               And [aip].[DatabaseName] = [ai].[DatabaseName]
                        Left Join [#ApSupplier] As [as]
                            On [as].[Supplier] = [ai].[Supplier]
                               And [as].[DatabaseName] = [ai].[DatabaseName]
                Where   [ai].[DueDate] <= GetDate()
                Group By [ai].[DatabaseName]
                      , [ai].[Supplier]
                      , [ai].[Invoice]
                      , [ai].[InvoiceDate]
                      , [ai].[DueDate]
                      , [ai].[OrigInvValue]
                      , [ai].[OrigDiscValue]
                      , [ai].[MthInvBal1]
                      , [ai].[MthInvBal2]
                      , [ai].[MthInvBal3]
                      , [as].[SupplierName]
                      , [ai].[ConvRate]
                      , [ai].[Currency]
                      , [ai].[Journal];

        --Print 6

        Insert  [#SupplierSummary]
                ( [DatabaseName]
                , [Supplier]
                , [SupplierValue]
                , [Currency]
                )
                Select  [r].[DatabaseName]
                      , [r].[Supplier]
                      , [SupplierValue] = Sum(Case When [r].[Value] >= 0
                                                   Then [r].[Value]
                                                   Else 0
                                              End)
                      , [r].[Currency]
                From    [#Results] As [r]
                Group By [r].[DatabaseName]
                      , [r].[Currency]
                      , [r].[Supplier];

--return results

        Select  [cn].[CompanyName]
              , [r].[Supplier]
              , [r].[SupplierName]
              , [SupplierValue] = Coalesce([ss].[SupplierValue] , 0)
              , [r].[Invoice]
              , [InvoiceDate] = Cast([r].[InvoiceDate] As Date)
              , [DueDate] = Cast([r].[DueDate] As Date)
```

Author: Johnson, Chris

```
                , [r].[OrigInvValue]
                , [r].[OrigDiscValue]
                , [r].[PaymentReference]
                , [r].[PaymentNumber]
                , [PostValue] = Coalesce([r].[PostValue] , 0)
                , [Value] = Coalesce([r].[Value] , 0)
                , [r].[MthInvBal1]
                , [r].[MthInvBal2]
                , [r].[MthInvBal3]
                , [r].[Currency]
                , [r].[ConvRate]
                , [r].[DatabaseName]
                , [r].[Journal]
        From    [#Results] [r]
                Left Join [Lookups].[CompanyNames] As [cn]
                    On [cn].[Company] = [r].[DatabaseName] Collate Latin1_General_-
BIN
                Left Join [#SupplierSummary] As [ss]
                    On [ss].[Supplier] = [r].[Supplier]
                       And [ss].[Currency] = [r].[Currency]
                       And [ss].[DatabaseName] = [r].[DatabaseName]
        Order By [DueDate];

    End;

GO
EXEC sp_addextendedproperty N'MS_Description', N'details of Ap Invoices including
payment details', 'SCHEMA', N'Report', 'PROCEDURE', N'UspResults_ApInvoicesWith-
PaymentDetails', NULL, NULL
GO
```

Uses

[Lookups].[CompanyNames]
[Process].[ExecForEachDB_WithTableCheck]
[Process].[UspInsert_RedTagLogs]
[Report]

Author: Johnson, Chris

[Report].[UspResults_ApJnlGLDistrs]

MS_Description

details of the general ledger distribution from accounts payable with journals

Parameters

Name	Data Type	Max Length (Bytes)
@Company	varchar(max)	max
@RedTagType	char	1
@RedTagUse	varchar(500)	500

SQL Script

```
CREATE Proc [Report].[UspResults_ApJnlGLDistrs]
    (
      @Company Varchar(Max)
    , @RedTagType Char(1)
    , @RedTagUse Varchar(500)
    )
As
    Begin
/*
Template designed by Chris Johnson, Prometic Group September 2015
Stored procedure set out to query multiple databases with the same information and
return it in a collated format
*/
        Set NoCount On;

        If IsNumeric(@Company) = 0
            Begin
                Select  @Company = Upper(@Company);
            End;

--Red tag

        Declare @RedTagDB Varchar(255)= Db_Name();
        Exec [Process].[UspInsert_RedTagLogs] @StoredProcDb = 'BlackBox' ,
            @StoredProcSchema = 'Report' ,
            @StoredProcName = 'UspResults_Template' ,
            @UsedByType = @RedTagType , @UsedByName = @RedTagUse ,
            @UsedByDb = @RedTagDB;

--list the tables that are to be pulled back from each DB - if they are not found
the script will not be run against that db

        Declare @ListOfTables Varchar(Max) = 'ApJnlSummary,ApJnlDistrib';

--create temporary tables to be pulled from different databases, including a column
to id
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_Ap-JnlGLDistrs

```
        Create Table [#ApJnlDists]
            (
              [DatabaseName] Varchar(150) Collate Latin1_General_BIN
            , [Invoice] Varchar(20) Collate Latin1_General_BIN
            , [Supplier] Varchar(15) Collate Latin1_General_BIN
            , [TrnYear] Int
            , [TrnMonth] Int
            , [Journal] Int
            , [EntryNumber] Int
            , [SubEntry] Int
            , [SupplierGlCode] Varchar(35) Collate Latin1_General_BIN
            , [TrnDate] Date
            , [ExpenseGlCode] Varchar(35) Collate Latin1_General_BIN
            , [DistrValue] Numeric(20 , 3)
            , [Reference] Varchar(30) Collate Latin1_General_BIN
            );



--create script to pull data from each db into the tables

        Declare @SQL Varchar(Max) = 'USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            BEGIN
                Insert  [#ApJnlDists]
                        ( [DatabaseName]
                        , [Invoice]
                        , [Supplier]
                        , [TrnYear]
                        , [TrnMonth]
                        , [Journal]
                        , [EntryNumber]
                        , [SubEntry]
                        , [SupplierGlCode]
                        , [TrnDate]
                        , [ExpenseGlCode]
                        , [DistrValue]
                        , [Reference]
                        )
                Select  @DBCode
                        , [AJS].[Invoice]
                        , [AJS].[Supplier]
                        , [AJS].[TrnYear]
                        , [AJS].[TrnMonth]
                        , [AJS].[Journal]
                        , [AJS].[EntryNumber]
                        , [AJD].[SubEntry]
                        , [AJS].[SupplierGlCode]
                        , [AJS].[TrnDate]
                        , [ExpenseGlCode] = Case When [AJD].[ExpenseGlCode] = ''''
                                                 Then Null
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_ApJnlGLDistrs

```
                                                Else [AJD].[ExpenseGlCode]
                                                End
                        , [AJD].[DistrValue]
                        , [Reference] = Case When [AJD].[Reference] = ''''
                                                Then Null
                                                Else [AJD].[Reference]
                                                End
                From    [dbo].[ApJnlSummary] [AJS]
                        Inner Join [dbo].[ApJnlDistrib] [AJD]
                            On [AJD].[TrnYear] = [AJS].[TrnYear]
                               And [AJD].[TrnMonth] = [AJS].[TrnMonth]
                               And [AJD].[Journal] = [AJS].[Journal]
                               And [AJD].[EntryNumber] = [AJS].[EntryNumber]
                               And [AJD].[AnalysisEntry] = [AJS].[AnalysisEntry];
            End
    End';

--Enable this function to check script changes (try to run script directly against
db manually)

--Print @SQL


--execute script against each db, populating the base tables

        Exec [Process].[ExecForEachDB_WithTableCheck] @cmd = @SQL ,
            @SchemaTablesToCheck = @ListOfTables;


--define the results you want to return


--Placeholder to create indexes as required


--script to combine base data and insert into results table



        Set NoCount Off;
--return results

        Select  [AJD].[DatabaseName]
              , [AJD].[Invoice]
              , [AJD].[Supplier]
              , [AJD].[TrnYear]
              , [AJD].[TrnMonth]
              , [AJD].[Journal]
              , [AJD].[EntryNumber]
              , [AJD].[SubEntry]
              , [AJD].[SupplierGlCode]
              , [AJD].[TrnDate]
              , [AJD].[ExpenseGlCode]
              , [AJD].[DistrValue]
              , [AJD].[Reference]
        From    [#ApJnlDists] [AJD];
```

Author: Johnson, Chris

```
    End;


GO
EXEC sp_addextendedproperty N'MS_Description', N'details of the general ledger
distribution from accounts payable with journals', 'SCHEMA', N'Report', 'PROCEDURE',
N'UspResults_ApJnlGLDistrs', NULL, NULL
GO
```

Uses

[Process].[ExecForEachDB_WithTableCheck]
[Process].[UspInsert_RedTagLogs]
[Report]

[Report].[UspResults_ApSupplierNar]

MS_Description

used to return AP supp narrative in documents

Parameters

Name	Data Type	Max Length (Bytes)
@Company	varchar(max)	max
@Supplier	varchar(max)	max

SQL Script

```
CREATE Proc [Report].[UspResults_ApSupplierNar]
    (
      @Company Varchar(Max)
    , @Supplier Varchar(Max)
    )
As
    Begin
/*
Template designed by Chris Johnson, Prometic Group September 2015
Stored procedure set out to query multiple databases with the same information and
return it in a collated format
Returns ApSupplierNar table for PO
*/
        Set NoCount Off;
        If IsNumeric(@Company) = 0
            Begin
                Select  @Company = Upper(@Company);
            End;

--remove nocount on to speed up query

        Set NoCount On;

--list the tables that are to be pulled back from each DB - if they are not found
the script will not be run against that db

        Declare @ListOfTables Varchar(Max) = 'ApSupplierNar';

--create temporary tables to be pulled from different databases, including a column
to id

        Create Table [#ApSupplierNar]
            (
              [DatabaseName] Varchar(150) Collate Latin1_General_BIN
            , [Supplier] Varchar(15) Collate Latin1_General_BIN
            , [Invoice] Varchar(20) Collate Latin1_General_BIN
            , [NoteType] Char(1) Collate Latin1_General_BIN
            , [Line] Int
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_Ap-SupplierNar

```
            , [Text] Varchar(100) Collate Latin1_General_BIN
            );


--create script to pull data from each db into the tables

        Declare @SQLApSupplierNar Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end'
            + --Only query DBs beginning SysProCompany

            '
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN'
            + --only companies selected in main run, or if companies selected then
all

            '
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT'
            + --count number of tables requested (number of commas plus one)

            '
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')'
            + --Count of the tables requested how many exist in the db

            '
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'','')) '
            + --only if the count matches (all the tables exist in the requested db)
then run the script

            '
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
                Insert #ApSupplierNar
                        ( DatabaseName
                        , Supplier
                        , Invoice
                        , NoteType
                        , Line
                        , Text
                        )
                SELECT DatabaseName = @DBCode
                     , Supplier
                     , Invoice
                     , NoteType
                     , Line
                     , Text
                FROM ApSupplierNar
                Where Supplier=''' + @Supplier + '''
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_Ap-SupplierNar

```
            End
    End';

--Enable this function to check script changes (try to run script directly against
db manually)

       -- Print @SQLApSupplierNar;


--execute script against each db, populating the base tables

        Exec [Process].[ExecForEachDB] @cmd = @SQLApSupplierNar;

--define the results you want to return


        Create Table [#Results]
            (
              [DatabaseName] Varchar(150)
            , [Supplier] Varchar(15)
            , [Invoice] Varchar(20)
            , [NoteType] Char(1)
            , [Line] Int
            , [Text] Varchar(100)
            );
--Placeholder to create indexes as required


--script to combine base data and insert into results table

        Insert  [#Results]
                ( [DatabaseName]
                , [Supplier]
                , [Invoice]
                , [NoteType]
                , [Line]
                , [Text]
                )
                Select  [Company] = [ASN].[DatabaseName]
                      , [ASN].[Supplier]
                      , [ASN].[Invoice]
                      , [ASN].[NoteType]
                      , [ASN].[Line]
                      , [ASN].[Text]
                From    [#ApSupplierNar] As [ASN];
--return results


        Select  [R].[DatabaseName]
              , [R].[Supplier]
              , [R].[Invoice]
              , [R].[NoteType]
              , [R].[Line]
              , [R].[Text]
        From    [#Results] As [R];

    End;
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_Ap-SupplierNar

```
GO
EXEC sp_addextendedproperty N'MS_Description', N'used to return AP supp narrative in documents', 'SCHEMA', N'Report', 'PROCEDURE', N'UspResults_ApSupplierNar', NULL, NULL
GO
```

Uses

[Process].[ExecForEachDB]
[Report]

Author: Johnson, Chris

[Report].[UspResults_ApSuppNarr]

MS_Description

used to return AP supp narrative in documents

Parameters

Name	Data Type	Max Length (Bytes)
@Company	varchar(max)	max
@RedTagType	char	1
@RedTagUse	varchar(500)	500

SQL Script

```
CREATE Proc [Report].[UspResults_ApSuppNarr]
    (
      @Company Varchar(Max)
    , @RedTagType Char(1)
    , @RedTagUse Varchar(500)
    )
As
    Begin
        Set NoCount On;

        If IsNumeric(@Company) = 0
            Begin
                Select  @Company = Upper(@Company);
            End;

--Red tag

        Declare @RedTagDB Varchar(255)= Db_Name();
        Exec [Process].[UspInsert_RedTagLogs] @StoredProcDb = 'BlackBox' ,
            @StoredProcSchema = 'Report' ,
            @StoredProcName = 'UspResults_ApSuppNarr' ,
            @UsedByType = @RedTagType , @UsedByName = @RedTagUse ,
            @UsedByDb = @RedTagDB;

--list the tables that are to be pulled back from each DB - if they are not found
the script will not be run against that db

        Declare @ListOfTables Varchar(Max) = 'ApSupplierNar';


        Create Table [#ApSupplierNar]
            (
              [DatabaseName] Varchar(150)    collate Latin1_General_BIN
            , [Supplier] Varchar(15)        collate Latin1_General_BIN
            , [Line] BigInt
            , [Text] Varchar(100)            collate Latin1_General_BIN
            );
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_Ap-SuppNarr

```
        --create script to pull data from each db into the tables

        Declare @SQL Varchar(Max) = 'USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            BEGIN
            Insert [#ApSupplierNar]
                    ( [DatabaseName]
                    , [Supplier]
                    , [Line]
                    , [Text]
                    )
            SELECT [DatabaseName]=@DBCode
                 , [ASN].[Supplier]
                 , [ASN].[Line]
                 , [ASN].[Text] FROM [ApSupplierNar] [ASN]
            End
    End';

--execute script against each db, populating the base tables

        Exec [Process].[ExecForEachDB_WithTableCheck] @cmd = @SQL ,
            @SchemaTablesToCheck = @ListOfTables;

        Set NoCount Off;

        Select  [ASN].[DatabaseName]
              , [ASN].[Supplier]
              , [ASN].[Line]
              , [ASN].[Text]
        From    [#ApSupplierNar] [ASN];


    End;

GO
EXEC sp_addextendedproperty N'MS_Description', N'used to return AP supp narrative in
documents', 'SCHEMA', N'Report', 'PROCEDURE', N'UspResults_ApSuppNarr', NULL, NULL
GO
```

Uses

[Process].[ExecForEachDB_WithTableCheck]
[Process].[UspInsert_RedTagLogs]
[Report]

Author: Johnson, Chris

[Report].[UspResults_ArCustomers]

MS_Description

list of all AR customers

Parameters

Name	Data Type	Max Length (Bytes)
@Company	varchar(max)	max
@RedTagType	char	1
@RedTagUse	varchar(500)	500

SQL Script

```
CREATE Proc [Report].[UspResults_ArCustomers]
    (
      @Company Varchar(Max)
    , @RedTagType Char(1)
    , @RedTagUse Varchar(500)
    )
As
    Begin
/*
Template designed by Chris Johnson, Prometic Group September 2015
Stored procedure set out to query multiple databases with the same information and
return it in a collated format
List of all AR customers
--exec [Report].[UspResults_ArCustomers] 10
*/

        If IsNumeric(@Company) = 0
            Begin
                Select  @Company = Upper(@Company);
            End;

--remove nocount on to speed up query

        Set NoCount On;

--Red tag

        Declare @RedTagDB Varchar(255)= Db_Name();
        Exec [Process].[UspInsert_RedTagLogs] @StoredProcDb = 'BlackBox' ,
            @StoredProcSchema = 'Report' ,
            @StoredProcName = 'UspResults_ArCustomers' ,
            @UsedByType = @RedTagType , @UsedByName = @RedTagUse ,
            @UsedByDb = @RedTagDB;

--list the tables that are to be pulled back from each DB - if they are not found
the script will not be run against that db
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_Ar-Customers

```
        Declare @ListOfTables Varchar(Max) = 'ArCustomer';

--create temporary tables to be pulled from different databases, including a column to id

        Create Table [#ArCustomer]
            (
              [DatabaseName] Varchar(150)    collate latin1_general_bin
            , [Customer] Varchar(25)        collate latin1_general_bin
            , [Name] Varchar(255)            collate latin1_general_bin
            , [InvoiceCount] Int
            , [DateLastSale] DateTime2
            , [DateLastPay] DateTime2
            , [DateCustAdded] DateTime2
            , [Contact] Varchar(100)        collate latin1_general_bin
            , [Telephone] Varchar(20)        collate latin1_general_bin
            , [Email] Varchar(255)            collate latin1_general_bin
            , [Nationality] Char(3)            collate latin1_general_bin
            );



--create script to pull data from each db into the tables

        Declare @SQL Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_Name(),len(db_Name())-13) else null end'
            + --Only query DBs beginning SysProCompany

            '
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN'
            + --only companies selected in main run, or if companies selected then all

            '
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT'
            + --count number of tables requested (number of commas plus one)

            '
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-String](@ListOfTables,'','')'
            + --Count of the tables requested how many exist in the db

            '
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-Box.dbo.udf_SplitString(@ListOfTables,'','')) '
            + --only if the count matches (all the tables exist in the requested db) then run the script

            '
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_Ar-Customers

```
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
                Insert [#ArCustomer]
        ( [DatabaseName]
        , [Customer]
        , [Name]
        , [InvoiceCount]
        , [DateLastSale]
        , [DateLastPay]
        , [DateCustAdded]
        , [Contact]
        , [Telephone]
        , [Email]
        , [Nationality]
        )
Select @DBCode
    ,[Customer]
    ,[Name]
    ,[InvoiceCount]
    ,[DateLastSale]
    ,[DateLastPay]
    ,[DateCustAdded]
    ,[Contact]
    ,[Telephone]
    ,[Email]
    ,[Nationality]
 FROM [dbo].[ArCustomer] As [ac]
            End
    End';

--Enable this function to check script changes (try to run script directly against db manually)

--Print @SQL


--execute script against each db, populating the base tables

        Exec [Process].[ExecForEachDB] @cmd = @SQL;

--define the results you want to return


--Placeholder to create indexes as required


--script to combine base data and insert into results table


--return results

        Select  [cn].[CompanyName]
              , [ac].[Customer]
              , [ac].[Name]
              , [ac].[InvoiceCount]
              , [DateLastSale] = Cast([ac].[DateLastSale] As Date)
              , [DateLastPay] = Cast([ac].[DateLastPay] As Date)
```

Author: Johnson, Chris

```
              , [DateCustAdded] = Cast([ac].[DateCustAdded] As Date)
              , [Contact] = Case When [ac].[Contact] = '' Then Null
                                 Else [ac].[Contact]
                            End
              , [ac].[Telephone]
              , [ac].[Email]
              , [ac].[Nationality]
        From    [#ArCustomer] As [ac]
                Left Join [BlackBox].[Lookups].[CompanyNames] As [cn] On
[ac].[DatabaseName] = [cn].[Company] Collate Latin1_General_BIN;

    End;

GO
EXEC sp_addextendedproperty N'MS_Description', N'list of all AR customers',
'SCHEMA', N'Report', 'PROCEDURE', N'UspResults_ArCustomers', NULL, NULL
GO
```

Uses

[Lookups].[CompanyNames]
[Process].[ExecForEachDB]
[Process].[UspInsert_RedTagLogs]
[Report]

Author: Johnson, Chris

[Report].[UspResults_ARDaysToPayment]

MS_Description

returns details on how long it takes for AR to be settled

Parameters

Name	Data Type	Max Length (Bytes)
@Company	varchar(max)	max
@RedTagType	char	1
@RedTagUse	varchar(500)	500

SQL Script

```
CREATE Proc [Report].[UspResults_ARDaysToPayment]
    (
      @Company Varchar(Max)
    , @RedTagType Char(1)
    , @RedTagUse Varchar(500)
    )
As
    Begin
/*
Template designed by Chris Johnson, Prometic Group September 2015
Stored procedure set out to query multiple databases with the same information and
return it in a collated format
*/
        If IsNumeric(@Company) = 0
            Begin
                Select  @Company = Upper(@Company);
            End;

--remove nocount on to speed up query

        Set NoCount On;

--Red tag

        Declare @RedTagDB Varchar(255)= Db_Name();
        Exec [Process].[UspInsert_RedTagLogs] @StoredProcDb = 'BlackBox' ,
            @StoredProcSchema = 'Report' ,
            @StoredProcName = 'UspResults_ARDaysToPayment' ,
            @UsedByType = @RedTagType , @UsedByName = @RedTagUse ,
            @UsedByDb = @RedTagDB;

--list the tables that are to be pulled back from each DB - if they are not found
the script will not be run against that db

        Declare @ListOfTables Varchar(Max) = 'AssetDepreciation,TblApTerms';

--create temporary tables to be pulled from different databases, including a column
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-ARDaysToPayment

```
to id

        Create Table [#ArInvoice]
            (
              [DatabaseName] Varchar(150)         Collate Latin1_General_BIN
            , [Customer] Varchar(15)             Collate Latin1_General_BIN
            , [SalesOrder] Varchar(20)             Collate Latin1_General_BIN
            , [Invoice] Varchar(20)                Collate Latin1_General_BIN
            , [InvoiceDate] Date
            , [CurrencyValue] Numeric(20 , 2)
            , [ConvRate] Numeric(20 , 6)
            , [MulDiv] Char(1)                     Collate Latin1_General_BIN
            , [PostCurrency] Varchar(3)            Collate Latin1_General_BIN
            , [DueDate] Date
            );
        Create Table [#ArInvoicePay]
            (
              [DatabaseName] Varchar(150)         Collate Latin1_General_BIN
            , [JournalDate] Date
            , [TrnValue] Numeric(20 , 2)
            , [InvoiceNotation] Varchar(50)        Collate Latin1_General_BIN
            , [PostCurrency] Varchar(3)            Collate Latin1_General_BIN
            , [PostConvRate] Numeric(20 , 6)
            , [PostMulDiv] Char(1)                 Collate Latin1_General_BIN
            , [Journal] Int
            , [Customer] Varchar(15)             Collate Latin1_General_BIN
            , [TrnType] Char(1)                    Collate Latin1_General_BIN
            , [Invoice] Varchar(20)                Collate Latin1_General_BIN
            );
        Create Table [#CshArPayments]
            (
              [DatabaseName] Varchar(150)         Collate Latin1_General_BIN
            , [CbTrnDate] Date
            , [GrossPayment] Numeric(20 , 2)
            , [NetPayment] Numeric(20 , 2)
            , [Customer] Varchar(15)             Collate Latin1_General_BIN
            , [Invoice] Varchar(20)                Collate Latin1_General_BIN
            , [Journal] Int
            , [Bank] Varchar(15)                 Collate Latin1_General_BIN
            );
        Create Table [#ApBank]
            (
              [DatabaseName] Varchar(150)         Collate Latin1_General_BIN
            , [Description] Varchar(50)            Collate Latin1_General_BIN
            , [Bank] Varchar(15)                 Collate Latin1_General_BIN
            );



--create script to pull data from each db into the tables

        Declare @SQL Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-ARDaysToPayment

```
    BEGIN
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables
            + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'',''))
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
                Insert [#ApBank]
                        ( [DatabaseName]
                        , [Description]
                        , [Bank]
                        )
                SELECT [DatabaseName]=@DBCode
                     , [AB].[Description]
                     , [AB].[Bank] FROM [ApBank] [AB]

                Insert [#ArInvoice]
                        ( [DatabaseName]
                        , [Customer]
                        , [SalesOrder]
                        , [Invoice]
                        , [InvoiceDate]
                        , [CurrencyValue]
                        , [ConvRate]
                        , [MulDiv]
                        , [PostCurrency]
                        , [DueDate]
                        )
                SELECT [DatabaseName]=@DBCode
                     , [AI].[Customer]
                     , [AI].[SalesOrder]
                     , [AI].[Invoice]
                     , [AI].[InvoiceDate]
                     , [AI].[CurrencyValue]
                     , [AI].[ConvRate]
                     , [AI].[MulDiv]
                     , [AI].[PostCurrency]
                     , DueDate = DateAdd(Day,[TAT].[DueDays],[AI].[InvoiceDate])
                From    [dbo].[ArInvoice] [AI]
                        Left Join [dbo].[TblArTerms] [TAT]
                            On [TAT].[TermsCode] = [AI].[TermsCode];

                Insert [#ArInvoicePay]
                        ( [DatabaseName]
                        , [JournalDate]
                        , [TrnValue]
                        , [InvoiceNotation]
                        , [PostCurrency]
                        , [PostConvRate]
                        , [PostMulDiv]
```

Author: Johnson, Chris

```
                        , [Journal]
                        , [Customer]
                        , [TrnType]
                        , [Invoice]
                        )
                SELECT [DatabaseName]=@DBCode
                     , [AIP].[JournalDate]
                     , [AIP].[TrnValue]
                     , [AIP].[InvoiceNotation]
                     , [AIP].[PostCurrency]
                     , [AIP].[PostConvRate]
                     , [AIP].[PostMulDiv]
                     , [AIP].[Journal]
                     , [AIP].[Customer]
                     , [AIP].[TrnType]
                     , [AIP].[Invoice] FROM [ArInvoicePay] [AIP]

                Insert [#CshArPayments]
                        ( [DatabaseName]
                        , [CbTrnDate]
                        , [GrossPayment]
                        , [NetPayment]
                        , [Customer]
                        , [Invoice]
                        , [Journal]
                        , [Bank]
                        )
                SELECT [DatabaseName]=@DBCode
                     , [CAP].[CbTrnDate]
                     , [CAP].[GrossPayment]
                     , [CAP].[NetPayment]
                     , [CAP].[Customer]
                     , [CAP].[Invoice]
                     , [CAP].[Journal]
                     , [CAP].[Bank] FROM [CshArPayments] [CAP]
            End
    End';

--execute script against each db, populating the base tables

        Exec [Process].[ExecForEachDB] @cmd = @SQL;

--define the results you want to return

        Create Table [#Results]
            (
              [Customer] Varchar(15)                Collate Latin1_General_BIN
            , [SalesOrder] Varchar(20)                Collate Latin1_General_BIN
            , [Invoice] Varchar(20)                   Collate Latin1_General_BIN
            , [InvoiceDate] Date
            , [DueDate] Date
            , [CurrencyValue] Numeric(20 , 2)
            , [ConvRate] Numeric(20 , 6)
            , [MulDiv] Varchar(1)                     Collate Latin1_General_BIN
            , [PostCurrency] Varchar(3)               Collate Latin1_General_BIN
            , [LocalValue] Numeric(20 , 2)
            , [JournalDate] Date
```

Author: Johnson, Chris

```
            , [TrnTypeDesc] Varchar(250)            Collate Latin1_General_BIN
            , [TrnValue] Numeric(20 , 2)
            , [InvoiceNotation] Varchar(50)           Collate Latin1_General_BIN
            , [JournalPostCurrency] Varchar(3)        Collate Latin1_General_BIN
            , [PostConvRate] Numeric(20 , 6)
            , [PostMulDiv] Varchar(1)
            , [LocalJournalValue] Numeric(20 , 2)
            , [Journal] Int
            , [CbTrnDate] Date
            , [GrossPayment] Numeric(20 , 2)
            , [NetPayment] Numeric(20 , 2)
            , [Bank] Varchar(50)                    Collate Latin1_General_BIN
            , [Company] Varchar(150)                Collate Latin1_General_BIN
            , [CompanyName] Varchar(250)            Collate Latin1_General_BIN
            );

--Placeholder to create indexes as required


--script to combine base data and insert into results table

        Insert  [#Results]
                ( [Customer]
                , [SalesOrder]
                , [Invoice]
                , [InvoiceDate]
                , [DueDate]
                , [CurrencyValue]
                , [ConvRate]
                , [MulDiv]
                , [PostCurrency]
                , [LocalValue]
                , [JournalDate]
                , [TrnTypeDesc]
                , [TrnValue]
                , [InvoiceNotation]
                , [JournalPostCurrency]
                , [PostConvRate]
                , [PostMulDiv]
                , [LocalJournalValue]
                , [Journal]
                , [CbTrnDate]
                , [GrossPayment]
                , [NetPayment]
                , [Bank]
                , [Company]
                , [CompanyName]
                )
                Select  [AI].[Customer]
                      , [AI].[SalesOrder]
                      , [AI].[Invoice]
                      , [AI].[InvoiceDate]
                      , [AI].[DueDate]
                      , [AI].[CurrencyValue]
                      , [AI].[ConvRate]
                      , [AI].[MulDiv]
                      , [AI].[PostCurrency]
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-ARDaysToPayment

```
                      , [LocalValue] = Case When [AI].[MulDiv] = 'D'
                                           Then Convert(Numeric(20 , 2) ,
[AI].[CurrencyValue]
                                                / [AI].[ConvRate])
                                           Else Convert(Numeric(20 , 2) ,
[AI].[CurrencyValue]
                                                * [AI].[ConvRate])
                                      End
                      , [JournalDate] = Convert(Date , [AIP].[JournalDate])
                      , [AIPTT].[TrnTypeDesc]
                      , [AIP].[TrnValue]
                      , [AIP].[InvoiceNotation]
                      , [JournalPostCurrency] = [AIP].[PostCurrency]
                      , [AIP].[PostConvRate]
                      , [AIP].[PostMulDiv]
                      , [LocalJournalValue] = Case When [AIP].[PostMulDiv] = 'D'
                                                   Then Convert(Numeric(20 , 2) ,
[AIP].[TrnValue]
                                                        / [AIP].[PostConvRate])
                                                   Else Convert(Numeric(20 , 2) ,
[AIP].[TrnValue]
                                                        * [AIP].[PostConvRate])
                                              End
                      , [AIP].[Journal]
                      , [CAP].[CbTrnDate]
                      , [CAP].[GrossPayment]
                      , [CAP].[NetPayment]
                      , [Bank] = [AB].[Description]
                      , [CN].[Company]
                      , [CN].[CompanyName]
                  From    [#ArInvoice] [AI]
                          Left Join [#ArInvoicePay] [AIP]
                              On [AIP].[Customer] = [AI].[Customer]
                                 And [AIP].[Invoice] = [AI].[Invoice]
                                 And [AIP].[DatabaseName] = [AI].[DatabaseName]
                          Left Join [BlackBox].[Lookups].[ArInvoicePayTrnType] [AIPTT]
                              On [AIPTT].[TrnType] = [AIP].[TrnType]
                          Left Join [#CshArPayments] [CAP]
                              On [CAP].[Customer] = [AIP].[Customer]
                                 And [CAP].[Invoice] = [AIP].[Invoice]
                                 And [CAP].[Journal] = [AIP].[Journal]
                                 And [CAP].[DatabaseName] = [AIP].[DatabaseName]
                          Left Join [#ApBank] [AB]
                              On [AB].[Bank] = [CAP].[Bank]
                                 And [AB].[DatabaseName] = [CAP].[DatabaseName]
                          Left Join [BlackBox].[Lookups].[CompanyNames] [CN]
                              On [CN].[Company] = [AI].[DatabaseName];

        Set NoCount Off;

        Select  [R].[InvoiceDate]
              , [R].[DueDate]
              , [R].[Invoice]
              , [R].[Customer]
              , [R].[LocalValue]
              , [R].[CurrencyValue]
              , [ReceivedValue] = Sum([R].[LocalJournalValue])
              , [LatestDate] = Max([R].[JournalDate])
```

Author: Johnson, Chris

```
              , [ClosedDate] = Case When Abs([R].[LocalValue]) <> Abs(Sum([R].[Local-
JournalValue]))
                                    Then Null
                                    Else Max([R].[JournalDate])
                               End
              , [DaysToCloseFromInvoiceDate] = DateDiff(Day ,
                                                        [R].[InvoiceDate] ,
                                                        Max([R].[JournalDate]))
              , [DaysToCloseFromDueDate] = DateDiff(Day , [R].[DueDate] ,
                                                    Max([R].[JournalDate]))
              , [R].[CompanyName]
        From    [#Results] [R]
        Group By [R].[InvoiceDate]
              , [R].[DueDate]
              , [R].[Invoice]
              , [R].[Customer]
              , [R].[LocalValue]
              , [R].[CurrencyValue]
              , [R].[CompanyName]
        Order By [R].[InvoiceDate] Desc;

    End;

GO
EXEC sp_addextendedproperty N'MS_Description', N'returns details on how long it
takes for AR to be settled', 'SCHEMA', N'Report', 'PROCEDURE', N'UspResults_ARDaysTo-
Payment', NULL, NULL
GO
```

Uses

[Lookups].[ArInvoicePayTrnType]
[Lookups].[CompanyNames]
[Process].[ExecForEachDB]
[Process].[UspInsert_RedTagLogs]
[Report]

Author: Johnson, Chris

[Report].[UspResults_AvailableLots]

MS_Description

list of all available lots

Parameters

Name	Data Type	Max Length (Bytes)
@Company	varchar(max)	max
@StockCode	varchar(50)	50
@Lot	varchar(1000)	1000
@IssueFromYN	char	1
@RedTagType	char	1
@RedTagUse	varchar(500)	500

SQL Script

```
CREATE Proc [Report].[UspResults_AvailableLots]
    (
      @Company Varchar(Max)
    , @StockCode Varchar(50)
    , @Lot Varchar(1000)
    , @IssueFromYN Char(1) --choose whether to only pick from warehouses that can
issue

    , @RedTagType Char(1)
    , @RedTagUse Varchar(500)
    )
As
    Begin
--remove nocount on to speed up query

        Set NoCount On;


        If IsNumeric(@Company) = 0
            Begin
                Select  @Company = Upper(@Company);
            End;


--If no stockcode selected, return all stock

        Select  @StockCode = Case When Lower(@StockCode) In ( '' , 'all' )
                                 Then 'ALL'
                                 Else Upper(@StockCode)
                            End;

--If no lot selected, return all lots
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-AvailableLots

```
        Select  @Lot = Case When Lower(@Lot) In ( '' , 'all' ) Then 'ALL'
                           Else Upper(@Lot)
                      End;

--Red tag

        Declare @RedTagDB Varchar(255)= Db_Name();
        Exec [Process].[UspInsert_RedTagLogs] @StoredProcDb = 'BlackBox' ,
            @StoredProcSchema = 'Report' ,
            @StoredProcName = 'UspResults_AvailableLots' ,
            @UsedByType = @RedTagType , @UsedByName = @RedTagUse ,
            @UsedByDb = @RedTagDB;

--list the tables that are to be pulled back from each DB - if they are not found
the script will not be run against that db

        Declare @ListOfTables Varchar(Max) = 'InvWhControl,InvMovements,InvMaster';

--create temporary tables to be pulled from different databases, including a column
to id

        Create Table [#InvWhControl]
            (
              [DatabaseName] Varchar(150)        collate latin1_general_bin
            , [Warehouse] Varchar(10)            collate latin1_general_bin
            , [Description] Varchar(50)            collate latin1_general_bin
            , [Fax] Varchar(20)                    collate latin1_general_bin
            );
        Create Table [#LotTransactions]
            (
              [DatabaseName] Varchar(150)        collate latin1_general_bin
            , [Warehouse] Varchar(10)            collate latin1_general_bin
            , [TrnType] Char(1)                    collate latin1_general_bin
            , [Reference] Varchar(30)            collate latin1_general_bin
            , [StockCode] Varchar(30)            collate latin1_general_bin
            , [TrnQuantity] Numeric(20 , 6)
            , [TimeStamp] NVarchar(50)             collate latin1_general_bin
            , [TrnDate] Date
            , [Lot] Varchar(50)                    collate latin1_general_bin
            , [EntryDateTime] As Left(Convert(NVarchar(50) , [TrnDate] , 121) ,
                                      10) + [TimeStamp]
            );
        Create Table [#InvMaster]
            (
              [DatabaseName] Varchar(150)        collate latin1_general_bin
            , [StockCode] Varchar(30)            collate latin1_general_bin
            , [Description] Varchar(50)            collate latin1_general_bin
            , [StockUom] Varchar(10)           collate latin1_general_bin
            , [Decimals] Int
            );

--create script to pull data from each db into the tables

        Declare @SQLInvWhControl Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-AvailableLots

```
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables
            + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'',''))
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
                Insert [#InvWhControl]
                        ( [DatabaseName]
                        , [Warehouse]
                        , [Description]
                        , [Fax]
                        )
                SELECT [DatabaseName]=@DBCode
                     , [IWC].[Warehouse]
                     , [IWC].[Description]
                     , [Fax] = case when upper([IWC].[Fax]) not in (''N'',''Y'')
then ''N'' else upper([IWC].[Fax]) end FROM [InvWhControl] [IWC]'
            + Case When @IssueFromYN = 'Y'
                   Then 'Where case when upper([IWC].[Fax]) not in (''N'',''Y'')
then ''N'' else upper([IWC].[Fax]) end=''Y'''
                   Else ''
              End + '            End
    End';
        Declare @SQLInvMaster Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables
            + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'',''))
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
                Insert [#InvMaster]
                    ( [DatabaseName]
                    , [StockCode]
                    , [Description]
                    , [StockUom]
                    , [Decimals]
                    )
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-AvailableLots

```
            SELECT [DatabaseName]=@DBCode
                 , [IM].[StockCode]
                 , [IM].[Description]
                 , [IM].[StockUom]
                 , [IM].[Decimals] FROM [InvMaster] [IM]'
            + Case When @StockCode <> 'ALL'
                   Then 'Where Upper([IM].[StockCode]) In (''' + @StockCode
                        + ''')'
                   Else ''
              End + '        End
    End';
        Declare @SQLLotTransactions Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables
            + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'',''))
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
                 Insert  [#LotTransactions]
                    ( [DatabaseName]
                    , [Warehouse]
                    , [Lot]
                    , [TrnType]
                    , [Reference]
                    , [StockCode]
                    , [TrnQuantity]
                    , [TimeStamp]
                    , [TrnDate]
                    )
                    Select  [DatabaseName] = @DBCode
                          , [LT].[Warehouse]
                          , [Lot]=substring(Lot, patindex(''%[^0]%'',Lot), 10)
                          , [LT].[TrnType]
                          , [LT].[Reference]
                          , [LT].[StockCode]
                          , [LT].[TrnQuantity]
                          , [TimeStamp] = Right(Convert(NVarchar(50) , Cast([Time-
Stamp] As DateTime) , 121) ,13)
                          , [LT].[TrnDate]
                    From    [LotTransactions] [LT]'
            + Case When @Lot <> 'ALL'
                        Or @StockCode <> 'ALL' Then ' where '
                   Else ''
              End
            + Case When @StockCode <> 'ALL'
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-AvailableLots

```
                    Then ' Upper([LT].[StockCode]) In (''' + @StockCode + ''')'
                    Else ''
               End + Case When @Lot <> 'ALL'
                               And @StockCode <> 'ALL' Then ' and '
                          Else ''
                     End
            + Case When @Lot <> 'ALL'
                    Then ' Upper(substring(Lot, patindex(''%[^0]%'',Lot), 10)) In
('''
                         + @Lot + ''')'
                    Else ''
               End + '
        End
    End';
--execute script against each db, populating the base tables

        Exec [Process].[ExecForEachDB] @cmd = @SQLInvWhControl;
        Exec [Process].[ExecForEachDB] @cmd = @SQLInvMaster;
        Exec [Process].[ExecForEachDB] @cmd = @SQLLotTransactions;


--define the results you want to return

        Create Table [#AllMovements]
            (
              [Warehouse] Varchar(10)                collate latin1_general_bin
            , [WarehouseDescription] Varchar(50)     collate latin1_general_bin
            , [IssueFrom] Varchar(20)                collate latin1_general_bin
            , [AmountModifier] Int
            , [TrnType] Char(1)                        collate latin1_general_bin
            , [Reference] Varchar(30)                collate latin1_general_bin
            , [Lot] Varchar(50)                        collate latin1_general_bin
            , [StockCode] Varchar(30)                collate latin1_general_bin
            , [StockDescription] Varchar(50)         collate latin1_general_bin
            , [StockUom] Varchar(10)                 collate latin1_general_bin
            , [TrnQty] Numeric(20 , 6)
            , [Decimals] Int
            , [EntryDateTime] DateTime2
            , [DescendingRank] BigInt
            , [DatabaseName] Varchar(150)            collate latin1_general_bin
            );

--Placeholder to create indexes as required

        --Create NonClustered Index [IM_Sc_Wh_Edt_Temp] On [#InvMovements] ([Stock-
Code],[Warehouse],[EntryDateTime]);

        Create NonClustered Index [Am_All_Temp] On [#AllMovements] ([Warehouse],
[IssueFrom], [StockCode], [StockUom], [DescendingRank]) Include ([Warehouse-
Description], [StockDescription], [Decimals]);

--script to combine base data and insert into results table

        Insert  [#AllMovements]
                ( [Warehouse]
                , [WarehouseDescription]
                , [IssueFrom]
                , [AmountModifier]
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-AvailableLots

```
                , [TrnType]
                , [Reference]
                , [Lot]
                , [StockCode]
                , [StockDescription]
                , [StockUom]
                , [TrnQty]
                , [Decimals]
                , [EntryDateTime]
                , [DescendingRank]
                , [DatabaseName]
                )
                Select  [IWC].[Warehouse]
                      , [WarehouseDescription] = [IWC].[Description]
                      , [IssueFrom] = [IWC].[Fax]
                      , [TTAM].[AmountModifier]
                      , [LT].[TrnType]
                      , [LT].[Reference]
                      , [LT].[Lot]
                      , [LT].[StockCode]
                      , [StockDescription] = [IM2].[Description]
                      , [IM2].[StockUom]
                      , [LT].[TrnQuantity]
                      , [IM2].[Decimals]
                      , [LT].[EntryDateTime]
                      , [DescendingRank] = Rank() Over ( Partition By [LT].[Stock-
Code] ,
                                                         [LT].[Warehouse] ,
                                                         [LT].[Lot] Order By
[LT].[EntryDateTime] Desc )
                      , [IWC].[DatabaseName]
                From    [#InvWhControl] [IWC]
                        Inner Join [#LotTransactions] [LT]
                            On [LT].[Warehouse] = [IWC].[Warehouse]
                               And [LT].[DatabaseName] = [IWC].[DatabaseName]
                        Left Join [#InvMaster] [IM2]
                            On [IM2].[StockCode] = [LT].[StockCode]
                               And [IM2].[DatabaseName] = [LT].[DatabaseName]
                        Left Join [BlackBox].[Lookups].[TrnTypeAmountModifier]
[TTAM]
                            On [TTAM].[TrnType] = [LT].[TrnType]
                               And [TTAM].[Company] = [LT].[DatabaseName];

        Set NoCount Off;

--return results

        Select  [AM].[Warehouse]
              , [AM].[WarehouseDescription]
              , [AM].[IssueFrom]
              , [AM].[Lot]
              , [AM].[StockCode]
              , [AM].[StockDescription]
              , [StockLevel] = Sum([AM].[AmountModifier] * [AM].[TrnQty])
              , [AM].[StockUom]
              , [AM].[Decimals]
              , [LastReference] = Max(Case When [AM].[DescendingRank] = 1
                                           Then [AM].[Reference]
```

Author: Johnson, Chris

```
                                        End)
              , [LastEntry] = Max(Case When [AM].[DescendingRank] = 1
                                       Then [AM].[EntryDateTime]
                                  End)
              , [LastEntryAmount] = Max(Case When [AM].[DescendingRank] = 1
                                             Then [AM].[TrnQty]
                                                  * [AM].[AmountModifier]
                                        End)
              , [LastEntryType] = Max(Case When [AM].[DescendingRank] = 1
                                           Then [AM].[TrnType]
                                      End)
              , [AM].[DatabaseName]
              , [CN].[CompanyName]
              , [CN].[ShortName]
        From    [#AllMovements] [AM]
                Left Join [Lookups].[CompanyNames] [CN]
                    On [AM].[DatabaseName] = [CN].[Company]
        Group By [AM].[Warehouse]
              , [AM].[WarehouseDescription]
              , [AM].[IssueFrom]
              , [AM].[Lot]
              , [AM].[StockCode]
              , [AM].[StockDescription]
              , [AM].[StockUom]
              , [AM].[Decimals]
              , [AM].[DatabaseName]
              , [CN].[CompanyName]
              , [CN].[ShortName];

    End;

GO
EXEC sp_addextendedproperty N'MS_Description', N'list of all available lots',
'SCHEMA', N'Report', 'PROCEDURE', N'UspResults_AvailableLots', NULL, NULL
GO
```

Uses

[Lookups].[CompanyNames]
[Lookups].[TrnTypeAmountModifier]
[Process].[ExecForEachDB]
[Process].[UspInsert_RedTagLogs]
[Report]

Author: Johnson, Chris

[Report].[UspResults_AvailableLots_AmountRequired]

MS_Description

list of available lots up to the required amount entered, lots picked from oldest to newest

Parameters

Name	Data Type	Max Length (Bytes)
@Company	varchar(10)	10
@StockCode	varchar(150)	150
@AmountRequired	numeric(20,8)	13
@RedTagType	char	1
@RedTagUse	varchar(500)	500

SQL Script

```
CREATE Proc [Report].[UspResults_AvailableLots_AmountRequired]
    (
      @Company Varchar(10)
    , @StockCode Varchar(150)
    , @AmountRequired Numeric(20 , 8)
    , @RedTagType Char(1)
    , @RedTagUse Varchar(500)
    )
As /*
Template designed by Chris Johnson, Prometic Group September 2015
Stored procedure set out to query multiple databases with the same information and
return it in a collated format
--Exec [Report].[UspResults_AvailableLots] @Company ='F', @StockCode
='000000000000013', @AmountRequired = 27.606400
*/
    Set NoCount On;

--Cater for if lower case companies are entered

    If IsNumeric(@Company) = 0
        Begin
            Select  @Company = Upper(@Company);
        End;

--Red tag

        Declare @RedTagDB Varchar(255)= Db_Name();
        Exec [Process].[UspInsert_RedTagLogs] @StoredProcDb = 'BlackBox' ,
            @StoredProcSchema = 'Report' ,
            @StoredProcName = 'UspResults_AvailableLots' ,
            @UsedByType = @RedTagType , @UsedByName = @RedTagUse ,
            @UsedByDb = @RedTagDB;

--list the tables that are to be pulled back from each DB - if they are not found
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-AvailableLots_AmountRequired

```
the script will not be run against that db

    Declare @ListOfTables Varchar(Max) = 'WipMasterSub,TblApTerms';

--Set maxmimum recursion to 9000

    Declare @CurrentJobLevel Int = 1
      , @TotalJobLevel Int= 9000
      , @InsertCount Int;

--Create table to capture results

    Create Table [#InvMaster]
        (
          [DatabaseName] Varchar(150)    Collate Latin1_General_BIN
        , [StockCode] Varchar(20)        Collate Latin1_General_BIN
        , [Description] Varchar(50)        Collate Latin1_General_BIN
        , [PartCategory] Char(1)        Collate Latin1_General_BIN
        , [IssMultLotsFlag] Char(1)        Collate Latin1_General_BIN
        , [StockUom] Varchar(10)        Collate Latin1_General_BIN
        );
    Create Table [#LotDetail]
        (
          [DatabaseName] Varchar(150)    Collate Latin1_General_BIN
        , [StockCode] Varchar(20)        Collate Latin1_General_BIN
        , [Lot] Varchar(20)                Collate Latin1_General_BIN
        , [Bin] Varchar(20)                Collate Latin1_General_BIN
        , [Warehouse] Varchar(30)        Collate Latin1_General_BIN
        , [QtyOnHand] Numeric(20 , 8)
        , [ExpiryDate] DateTime2
        , [CreationDate] DateTime2
        );


--create script to pull data from each db into the tables

    Declare @SQLInvMaster Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end'
        + --Only query DBs beginning SysProCompany

        '
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN'
        + --only companies selected in main run, or if companies selected then all

        '
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
        + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT'
        + --count number of tables requested (number of commas plus one)

        '
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-AvailableLots_AmountRequired

```
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')'
        + --Count of the tables requested how many exist in the db

        '
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'','')) '
        + --only if the count matches (all the tables exist in the requested db)
then run the script

        '
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
            Insert [#InvMaster]
                    ( [DatabaseName]
                    , [StockCode]
                    , [PartCategory]
                    , [IssMultLotsFlag]
                    , [StockUom]
                    , [Description]
                    )
            SELECT [DatabaseName]=@DBCode
                 , [StockCode]
                 , [PartCategory]
                 , [IssMultLotsFlag]
                 , [StockUom]
                 , [Description]
            FROM [InvMaster] As [im]
            where [StockCode] =''' + @StockCode + '''
            End
    End';
    Declare @SQLLotDetail Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end'
        + --Only query DBs beginning SysProCompany

        '
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN'
        + --only companies selected in main run, or if companies selected then all

        '
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
        + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT'
        + --count number of tables requested (number of commas plus one)

        '
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')'
        + --Count of the tables requested how many exist in the db

        '
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-AvailableLots_AmountRequired

```
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'','')) '
        + --only if the count matches (all the tables exist in the requested db)
then run the script

        '
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
                Insert [#LotDetail]
                        ( [DatabaseName]
                        , [StockCode]
                        , [Lot]
                        , [Bin]
                        , [Warehouse]
                        , [QtyOnHand]
                        , [ExpiryDate]
                        , [CreationDate]
                        )
                Select @DBCode
                  , [ld].[StockCode]
                  , [ld].[Lot]
                  , [ld].[Bin]
                  , [ld].[Warehouse]
                  , [ld].[QtyOnHand]
                  , [ld].[ExpiryDate]
                  , [ld].[CreationDate]
                From
                    [LotDetail] As [ld]
                Where
                    [ld].[QtyOnHand] <> 0
                    and [StockCode] =''' + @StockCode + '''
            End
    End';

    --Print 1

    Exec [Process].[ExecForEachDB] @cmd = @SQLInvMaster;
    --Print 2

    Exec [Process].[ExecForEachDB] @cmd = @SQLLotDetail;

    Create Table [#Results]
        (
          [StockCode] Varchar(20)                    Collate Latin1_General_BIN
        , [StockDescription] Varchar(50)            Collate Latin1_General_BIN
        , [IssMultLotsFlag] Char(1)                    Collate Latin1_General_BIN
        , [AvailableLot] Varchar(20)                Collate Latin1_General_BIN
        , [AvailableLotBin] Varchar(20)                Collate Latin1_General_BIN
        , [AvailableLotWarehouse] Varchar(30)       Collate Latin1_General_BIN
        , [AvailableLotQtyOnHand] Numeric(20 , 8)
        , [AvailableLotExpiryDate] Date
        , [AvailableLotCreationDate] Date
        , [RunningTotal] Numeric(20 , 8)
        , [LotRank] Int
        );
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-AvailableLots_AmountRequired

```
    Insert  [#Results]
            ( [StockCode]
            , [StockDescription]
            , [IssMultLotsFlag]
            , [AvailableLot]
            , [AvailableLotBin]
            , [AvailableLotWarehouse]
            , [AvailableLotQtyOnHand]
            , [AvailableLotExpiryDate]
            , [AvailableLotCreationDate]
            , [LotRank]
            )
            Select  [im].[StockCode]
                  , [StockDescription] = [im].[Description]
                  , [im].[IssMultLotsFlag]
                  , [AvailableLot] = [ld].[Lot]
                  , [AvailableLotBin] = [ld].[Bin]
                  , [AvailableLotWarehouse] = [ld].[Warehouse]
                  , [AvailableLotQtyOnHand] = [ld].[QtyOnHand]
                  , [AvailableLotExpiryDate] = [ld].[ExpiryDate]
                  , [AvailableLotCreationDate] = [ld].[CreationDate]
                  , [LotRank] = Dense_Rank() Over ( Partition By [ld].[StockCode]
Order By [ld].[CreationDate] Asc, [ld].[Lot] Asc )
            From    [#InvMaster] As [im]
                    Left Join [#LotDetail] As [ld] On [ld].[DatabaseName] =
[im].[DatabaseName]
                                                      And [ld].[StockCode] =
[im].[StockCode]
                                                      And [im].[IssMultLotsFlag] =
'Y';


    Declare @RunningTotal Numeric(20 , 8)= 0;

    Update  [#Results]
    Set     @RunningTotal = [RunningTotal] = @RunningTotal
            + [AvailableLotQtyOnHand]
    From    [#Results];

    Declare @Min Numeric(20 , 8)
      , @Max Numeric(20 , 8);

    Select  @Min = Min([R].[RunningTotal])
    From    [#Results] As [R]
    Where   [R].[RunningTotal] > @AmountRequired;

    Select  @Max = Max([R].[RunningTotal])
    From    [#Results] As [R]
    Where   [R].[RunningTotal] <= @AmountRequired;

    Print 1;
    Print @Min;
    Print 2;
    Print @Max;

    If @Min Is Null
        And @Max Is Null
        Begin
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-AvailableLots_AmountRequired

```
            Select  @AmountRequired = Min([R].[RunningTotal])
            From    [#Results] As [R];
        End;

    If Coalesce(@Max , 0) < @AmountRequired
        And @Min Is Not Null
        Begin
            Set @AmountRequired = @Min;
        End;

    Select  [R].[StockCode]
          , [R].[StockDescription]
          , [R].[IssMultLotsFlag]
          , [R].[AvailableLot]
          , [R].[AvailableLotBin]
          , [R].[AvailableLotWarehouse]
          , [R].[AvailableLotQtyOnHand]
          , [R].[AvailableLotExpiryDate]
          , [R].[AvailableLotCreationDate]
          , [R].[RunningTotal]
          , [R].[LotRank]
    From    [#Results] As [R]
            Left Join [#Results] As [R2] On [R2].[AvailableLot] = [R].[AvailableLot]
+ 1
    Where   [R].[RunningTotal] <= @AmountRequired;
    --Or R.LotRank=1;


--tidy up

    Drop Table [#InvMaster];
    Drop Table [#LotDetail];




GO
EXEC sp_addextendedproperty N'MS_Description', N'list of available lots up to the
required amount entered, lots picked from oldest to newest', 'SCHEMA', N'Report',
'PROCEDURE', N'UspResults_AvailableLots_AmountRequired', NULL, NULL
GO
```

Uses

[Process].[ExecForEachDB]
[Process].[UspInsert_RedTagLogs]
[Report]

Author: Johnson, Chris

[Report].[UspResults_AvailableStock]

MS_Description

list of stock available in each warehouse

Parameters

Name	Data Type	Max Length (Bytes)
@Company	varchar(max)	max
@StockCode	varchar(50)	50
@IssueFromYN	char	1
@RedTagType	char	1
@RedTagUse	varchar(500)	500

SQL Script

```
CREATE Proc [Report].[UspResults_AvailableStock]
    (
      @Company Varchar(Max)
    , @StockCode Varchar(50)
    , @IssueFromYN Char(1) --choose whether to only pick from warehouses that can
issue

    , @RedTagType Char(1)
    , @RedTagUse Varchar(500)
    )
As
    Begin
--remove nocount on to speed up query

        Set NoCount On;


        If IsNumeric(@Company) = 0
            Begin
                Select  @Company = Upper(@Company);
            End;


--If no stockcode selected, return all stock

        Select  @StockCode = Case When Lower(@StockCode) In ( '' , 'all' )
                                  Then 'ALL'
                                  Else Upper(@StockCode)
                             End;

--Red tag

        Declare @RedTagDB Varchar(255)= Db_Name();
        Exec [Process].[UspInsert_RedTagLogs] @StoredProcDb = 'BlackBox' ,
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-AvailableStock

```
            @StoredProcSchema = 'Report' ,
            @StoredProcName = 'UspResults_AvailableStock' ,
            @UsedByType = @RedTagType , @UsedByName = @RedTagUse ,
            @UsedByDb = @RedTagDB;

--list the tables that are to be pulled back from each DB - if they are not found
the script will not be run against that db

        Declare @ListOfTables Varchar(Max) = 'InvWhControl,InvMovements,InvMaster';

--create temporary tables to be pulled from different databases, including a column
to id

        Create Table [#InvWhControl]
            (
              [DatabaseName] Varchar(150)    collate latin1_general_bin
            , [Warehouse] Varchar(10)        collate latin1_general_bin
            , [Description] Varchar(50)        collate latin1_general_bin
            , [Fax] Varchar(20)                collate latin1_general_bin
            );
        Create Table [#InvMovements]
            (
              [DatabaseName] Varchar(150)    collate latin1_general_bin
            , [Warehouse] Varchar(10)        collate latin1_general_bin
            , [MovementType] Char(1)        collate latin1_general_bin
            , [TrnType] Char(1)                collate latin1_general_bin
            , [Reference] Varchar(30)        collate latin1_general_bin
            , [StockCode] Varchar(30)        collate latin1_general_bin
            , [TrnQty] Numeric(20 , 6)
            , [EntryDate] DateTime
            , [TrnTime] Int
            , [EntryDateTime] As DateAdd(Millisecond ,
                                         Convert(Int , Right([TrnTime] , 2)) ,
                                         DateAdd(Second ,
                                                 Convert(Int , Left(Right([TrnTime]
,
                                                                    4) , 2)) ,
                                                 DateAdd(Minute ,
                                                         Convert(Int ,
Left(Right([TrnTime] ,
                                                                    6) , 2)) ,
                                                         DateAdd(Hour ,
                                                              Convert(Int ,
Left([TrnTime] ,
                                                                    2)) ,
                                                              [EntryDate])))))
            );
        Create Table [#InvMaster]
            (
              [DatabaseName] Varchar(150)    collate latin1_general_bin
            , [StockCode] Varchar(30)        collate latin1_general_bin
            , [Description] Varchar(50)        collate latin1_general_bin
            , [StockUom] Varchar(10)        collate latin1_general_bin
            , [Decimals] Int
            );

--create script to pull data from each db into the tables
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-AvailableStock

```
        Declare @SQLInvWhControl Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables
            + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'',''))
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
                Insert [#InvWhControl]
                        ( [DatabaseName]
                        , [Warehouse]
                        , [Description]
                        , [Fax]
                        )
                SELECT [DatabaseName]=@DBCode
                     , [IWC].[Warehouse]
                     , [IWC].[Description]
                     , [Fax] = case when upper([IWC].[Fax]) not in (''N'',''Y'')
then ''N'' else upper([IWC].[Fax]) end FROM [InvWhControl] [IWC]'
            + Case When @IssueFromYN = 'Y'
                   Then 'Where case when upper([IWC].[Fax]) not in (''N'',''Y'')
then ''N'' else upper([IWC].[Fax]) end=''Y'''
                   Else ''
              End + '            End
    End';
        Declare @SQLInvMaster Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables
            + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'',''))
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
                Insert [#InvMaster]
                    ( [DatabaseName]
```

Author: Johnson, Chris

```
                    , [StockCode]
                    , [Description]
                    , [StockUom]
                    , [Decimals]
                    )
            SELECT [DatabaseName]=@DBCode
                 , [IM].[StockCode]
                 , [IM].[Description]
                 , [IM].[StockUom]
                 , [IM].[Decimals] FROM [InvMaster] [IM]'
            + Case When @StockCode <> 'ALL'
                   Then 'Where Upper([IM].[StockCode]) In (''' + @StockCode
                        + ''')'
                   Else ''
              End + '        End
    End';
        Declare @SQLInvMovements Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables
            + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'',''))
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
                Insert [#InvMovements]
                        ( [DatabaseName]
                        , [Warehouse]
                        , [MovementType]
                        , [TrnType]
                        , [Reference]
                        , [StockCode]
                        , [TrnQty]
                        , [EntryDate]
                        , [TrnTime]
                        )
                SELECT [DatabaseName]=@DBCode
                     , [IM].[Warehouse]
                     , [IM].[MovementType]
                     , [TrnType] = Case When [IM].[TrnType] = ''''
                                        Then [IM].[MovementType]
                                        Else [IM].[TrnType]
                                   End
                     , [IM].[Reference]
                     , [IM].[StockCode]
                     , [IM].[TrnQty]
                     , [IM].[EntryDate]
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-AvailableStock

```
                    , [IM].[TrnTime] FROM [InvMovements] [IM]'
           + Case When @StockCode <> 'ALL'
                  Then 'Where Upper([IM].[StockCode]) In (''' + @StockCode
                       + ''')'
                  Else ''
             End + '        End
    End';
--execute script against each db, populating the base tables

        Exec [Process].[ExecForEachDB] @cmd = @SQLInvWhControl;
        Exec [Process].[ExecForEachDB] @cmd = @SQLInvMaster;
        Exec [Process].[ExecForEachDB] @cmd = @SQLInvMovements;


--define the results you want to return

        Create Table [#AllMovements]
            (
              [Warehouse] Varchar(10)                collate latin1_general_bin
            , [WarehouseDescription] Varchar(50)    collate latin1_general_bin
            , [IssueFrom] Varchar(20)                collate latin1_general_bin
            , [MovementType] Char(1)                collate latin1_general_bin
            , [AmountModifier] Int
            , [TrnType] Char(1)                       collate latin1_general_bin
            , [Reference] Varchar(30)                collate latin1_general_bin
            , [StockCode] Varchar(30)                collate latin1_general_bin
            , [StockDescription] Varchar(50)        collate latin1_general_bin
            , [StockUom] Varchar(10)                collate latin1_general_bin
            , [TrnQty] Numeric(20 , 6)
            , [Decimals] Int
            , [EntryDateTime] DateTime2
            , [DescendingRank] BigInt
            );

--Placeholder to create indexes as required

        Create NonClustered Index [IM_Sc_Wh_Edt_Temp] On [#InvMovements] ([Stock-
Code],[Warehouse],[EntryDateTime]);
        Create NonClustered Index [Am_All_Temp] On [#AllMovements] ([Warehouse],
[IssueFrom], [StockCode], [StockUom], [DescendingRank]) Include ([Warehouse-
Description], [StockDescription], [Decimals]);

--script to combine base data and insert into results table

        Insert  [#AllMovements]
                ( [Warehouse]
                , [WarehouseDescription]
                , [IssueFrom]
                , [MovementType]
                , [AmountModifier]
                , [TrnType]
                , [Reference]
                , [StockCode]
                , [StockDescription]
                , [StockUom]
                , [TrnQty]
                , [Decimals]
                , [EntryDateTime]
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-AvailableStock

```
                , [DescendingRank]
                )
                Select  [IWC].[Warehouse]
                      , [WarehouseDescription] = [IWC].[Description]
                      , [IssueFrom] = [IWC].[Fax]
                      , [IM].[MovementType]
                      , [TTAM].[AmountModifier]
                      , [IM].[TrnType]
                      , [IM].[Reference]
                      , [IM].[StockCode]
                      , [StockDescription] = [IM2].[Description]
                      , [IM2].[StockUom]
                      , [IM].[TrnQty]
                      , [IM2].[Decimals]
                      , [IM].[EntryDateTime]
                      , [DescendingRank] = Rank() Over ( Partition By [IM].[Stock-
Code] ,
                                                                    [IM].[Warehouse] Order By
[IM].[EntryDateTime] Desc )
                From    [#InvWhControl] [IWC]
                        Inner Join [#InvMovements] [IM]
                            On [IM].[Warehouse] = [IWC].[Warehouse]
                               And [IM].[DatabaseName] = [IWC].[DatabaseName]
                        Left Join [#InvMaster] [IM2]
                            On [IM2].[StockCode] = [IM].[StockCode]
                               And [IM2].[DatabaseName] = [IM].[DatabaseName]
                        Left Join [BlackBox].[Lookups].[TrnTypeAmountModifier]
[TTAM]
                            On [TTAM].[TrnType] = [IM].[TrnType]
                               And [TTAM].[Company] = [IM].[DatabaseName];

        Set NoCount Off

--return results

        Select  [AM].[Warehouse]
              , [AM].[WarehouseDescription]
              , [AM].[IssueFrom]
              , [AM].[StockCode]
              , [AM].[StockDescription]
              , [StockLevel] = Sum([AM].[AmountModifier] * [AM].[TrnQty])
              , [AM].[StockUom]
              , [AM].[Decimals]
              , [LastReference] = Max(Case When [AM].[DescendingRank] = 1
                                           Then [AM].[Reference]
                                      End)
              , [LastEntry] = Max(Case When [AM].[DescendingRank] = 1
                                       Then [AM].[EntryDateTime]
                                  End)
              , [LastEntryAmount] = Max(Case When [AM].[DescendingRank] = 1
                                             Then [AM].[TrnQty]
                                                  * [AM].[AmountModifier]
                                        End)
              , [LastEntryType] = Max(Case When [AM].[DescendingRank] = 1
                                           Then [AM].[TrnType]
                                      End)
        From    [#AllMovements] [AM]
        Group By [AM].[Warehouse]
```

Author: Johnson, Chris

```
                , [AM].[WarehouseDescription]
                , [AM].[IssueFrom]
                , [AM].[StockCode]
                , [AM].[StockDescription]
                , [AM].[StockUom]
                , [AM].[Decimals];

    End;

GO
EXEC sp_addextendedproperty N'MS_Description', N'list of stock available in each warehouse', 'SCHEMA', N'Report', 'PROCEDURE', N'UspResults_AvailableStock', NULL, NULL
GO
```

Uses

[Lookups].[TrnTypeAmountModifier]
[Process].[ExecForEachDB]
[Process].[UspInsert_RedTagLogs]
[Report]

Author: Johnson, Chris

[Report].[UspResults_BudgetsActuals]

MS_Description

budgets vs actuals from GL

Parameters

Name	Data Type	Max Length (Bytes)
@RedTagType	char	1
@RedTagUse	varchar(500)	500

SQL Script

```
CREATE Proc [Report].[UspResults_BudgetsActuals]
    (
      @RedTagType Char(1)
    , @RedTagUse Varchar(500)
    )
As
    Begin
/*
Template designed by Chris Johnson, Prometic Group March 2016
*/

--remove nocount on to speed up query

        Set NoCount On;

--Red tag

        Declare @RedTagDB Varchar(255)= Db_Name();
        Exec [Process].[UspInsert_RedTagLogs] @StoredProcDb = 'BlackBox' ,
            @StoredProcSchema = 'Report' ,
            @StoredProcName = 'UspResults_BudgetsActuals' ,
            @UsedByType = @RedTagType , @UsedByName = @RedTagUse ,
            @UsedByDb = @RedTagDB;

        Create Table [#Budgets]
            (
              [Company] Varchar(10) Collate Latin1_General_BIN
            , [GlCode] Varchar(100) Collate Latin1_General_BIN
            , [Budget] Numeric(20 , 2)
            , [BudgetType] Char(1) Collate Latin1_General_BIN
            , [Period] Int
            );
        Create Table [#Actuals]
            (
              [Company] Varchar(10) Collate Latin1_General_BIN
            , [GlCode] Varchar(50) Collate Latin1_General_BIN
            , [Period] Int
            , [Actual] Numeric(20 , 2)
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-BudgetsActuals

```
            , [YearMovement] Numeric(20 , 2)
            , [GlYear] Int
            , [ClosingBalance] Numeric(20 , 2)
            , [MovementToDate] Numeric(20 , 2)
            );
        Create Table [#BudgetsActuals]
            (
              [CompanyGlCode] Varchar(150) Collate Latin1_General_BIN
            , [Period] Int
            , [Budget] Numeric(20 , 2)
            , [Actual] Numeric(20 , 2)
            , [GlYear] Int
            , [LineType] Varchar(50) Collate Latin1_General_BIN
            , [ClosingBalance] Numeric(20 , 2)
            , [MovementToDate] Numeric(20 , 2)
            , [Company] Varchar(10) Collate Latin1_General_BIN
            );

        Insert  [#Budgets]
                ( [Company]
                , [GlCode]
                , [Budget]
                , [BudgetType]
                , [Period]
                )
                Exec [BlackBox].[Report].[UspResults_GL_Budgets];
        Insert  [#Actuals]
                ( [Company]
                , [GlCode]
                , [Period]
                , [Actual]
                , [YearMovement]
                , [GlYear]
                , [ClosingBalance]
                , [MovementToDate]
                )
                Exec [Report].[UspResults_GL_Actuals];
        Insert  [#BudgetsActuals]
                ( [CompanyGlCode]
                , [Period]
                , [Budget]
                , [Actual]
                , [GlYear]
                , [LineType]
                , [ClosingBalance]
                , [MovementToDate]
                , [Company]
                )
                Select  [t].[CompanyGlCode]
                      , [t].[Period]
                      , [t].[Budget]
                      , [t].[Actual]
                      , [t].[GlYear]
                      , [t].[LineType]
                      , [t].[ClosingBalance]
                      , [t].[MovementToDate]
                      , [t].[Company]
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-BudgetsActuals

```
                From    ( Select    [CompanyGlCode] = [A].[GlCode]
                                  , [A].[Period]
                                  , [Budget] = 0
                                  , [A].[Actual]
                                  , [A].[GlYear]
                                  , [LineType] = 'Actual'
                                  , [A].[ClosingBalance]
                                  , [A].[MovementToDate]
                                  , [A].[Company]
                            From    [#Actuals] As [A]
                            Union
                            Select    [CompanyGlCode] = [B].[GlCode]
                                  , [B].[Period]
                                  , [B].[Budget]
                                  , [Actual] = 0
                                  , [GlYear] = Year(GetDate())
                                  , [LineType] = 'Budget'
                                  , [ClosingBalance] = 0
                                  , [MovementToDate] = 0
                                  , [B].[Company]
                            From    [#Budgets] As [B]
                          ) [t];

        Select  [BA].[CompanyGlCode]
              , [BA].[Period]
              , [BA].[Budget]
              , [BA].[Actual]
              , [BA].[GlYear]
              , [BA].[LineType]
              , [BA].[ClosingBalance]
              , [BA].[MovementToDate]
              , [BA].[Company]
              , [GM].[Description]
              , [GlGroup] = Case When [GM].[GlGroup] = '' Then Null
                                 Else [GM].[GlGroup]
                            End
              , [CN].[CompanyName]
              , [ReportIndex2] = Case When [GM].[ReportIndex2] = '' Then Null
                                      When [BA].[Company] = '10'
                                      Then 'Co. 10 ' + [GM].[ReportIndex2]
                                      Else [GM].[ReportIndex2]
                                 End
              , [CN].[Currency]
              , [CR].[CADMultiply]
              , [CR].[GBPMultiply]
              , [CR].[USDMultiply]
              , [CR].[EURMultiply]
              , [CR].[CHFMultiply]
              , [GAT].[GLAccountTypeDesc]
              , [RIUM].[Map]
              , [RIUM].[IsSummary]
              , [GroupMap1] = [LGM].[Map1]
              , [GroupMap2] = [LGM].[Map2]
              , [GroupMap3] = [LGM].[Map3]
        From    [#BudgetsActuals] As [BA]
                Left Join [SysproCompany40].[dbo].[GenMaster] As [GM] On
[GM].[Company] = [BA].[Company]
```

Author: Johnson, Chris

```
                                                                And [GM].[GlCode] =
[BA].[CompanyGlCode]
                Left Join [BlackBox].[Lookups].[CompanyNames] As [CN] On
[CN].[Company] = [BA].[Company]
                Left Join [Lookups].[CurrencyRates] As [CR] On [CR].[Currency] =
[CN].[Currency]
                                                                And GetDate() Between
[CR].[StartDateTime]
                                                                And
                                                                [CR].[EndDateTime]
                Left Join [BlackBox].[Lookups].[GLAccountType] As [GAT] On
[GM].[AccountType] = [GAT].[GLAccountType]
                Left Join [BlackBox].[Lookups].[LedgerGroupMaps] As [LGM] On
[LGM].[GlGroup] = [GM].[GlGroup]
                Left Join [Lookups].[ReportIndexUserMaps] As [RIUM] On
[RIUM].[ReportIndex2] = Case
                                                                When [GM].[Report-
Index2] = ''
                                                                Then Null
                                                                When [BA].[Company] =
'10'
                                                                Then 'Co. 10 '
                                                                + [GM].[ReportIndex2]
                                                                Else [GM].[Report-
Index2]
                                                                End;


    End;
GO
EXEC sp_addextendedproperty N'MS_Description', N'budgets vs actuals from GL',
'SCHEMA', N'Report', 'PROCEDURE', N'UspResults_BudgetsActuals', NULL, NULL
GO
```

Uses

[Lookups].[CompanyNames]
[Lookups].[CurrencyRates]
[Lookups].[GLAccountType]
[Lookups].[LedgerGroupMaps]
[Lookups].[ReportIndexUserMaps]
[Process].[UspInsert_RedTagLogs]
[Report].[UspResults_GL_Actuals]
[Report].[UspResults_GL_Budgets]
[Report]

Author: Johnson, Chris

[Report].[UspResults_BudgetsVsActuals]

MS_Description

budgets vs actuals from GL

SQL Script

```
CREATE Proc [Report].[UspResults_BudgetsVsActuals]
As /*Function designed by Chris Johnson, Prometic Group September 2015
Stored procedure to return details from General Ledger in Company 40 regarding
budget and actual figures*/


    Set NoCount On;

--unpivot budget data

    Create Table [#UnpivotBudget]
        (
          [Company] Varchar(10) Collate Latin1_General_BIN
        , [GlCode] Varchar(100) Collate Latin1_General_BIN
        , [BudgetPeriod] Int
        , [Budget] Numeric
        , [YearBudget] Numeric
        );

    Insert  [#UnpivotBudget]
            ( [Company]
            , [GlCode]
            , [BudgetPeriod]
            , [Budget]
            , [YearBudget]
            )
            Select Distinct
                    [ASMT].[Company]
                  , [ASMT].[GlCode]
                  , [BudgetPeriod] = Cast(Replace([ASMT].[BudgetPeriod] , 'Budget' ,
                                                  '') As Int)
                  , [ASMT].[Budget]
                  , [YearBudget] = [GB].[Budget1] + [GB].[Budget2] + [GB].[Budget3]
                    + [GB].[Budget4] + [GB].[Budget5] + [GB].[Budget6] +
[GB].[Budget7]
                    + [GB].[Budget8] + [GB].[Budget9] + [GB].[Budget10] +
[GB].[Budget11]
                    + [GB].[Budget12]
            From    [SysproCompany40].[dbo].[GenBudgets] Unpivot ( [Budget] For
[BudgetPeriod] In ( [Budget1] ,
                                                                   [Budget2] ,
                                                                   [Budget3] ,
                                                                   [Budget4] ,
                                                                   [Budget5] ,
                                                                   [Budget6] ,
                                                                   [Budget7] ,
```

Author: Johnson, Chris

```
                                                                [Budget8] ,
                                                                [Budget9] ,
                                                                [Budget10] ,
                                                                [Budget11] ,
                                                                [Budget12] ) ) As
[ASMT]
                    Left Join [SysproCompany40].[dbo].[GenBudgets] [GB]
                        On [GB].[Company] = [ASMT].[Company]
                           And [GB].[BudgetNumber] = [ASMT].[BudgetNumber]
                           And [GB].[GlCode] = [ASMT].[GlCode]
            Where   [ASMT].[BudgetType] = 'C';--Current budget only




--create list of actuals

    Create Table [#Actuals]
        (
          [Company] Varchar(10)            collate latin1_general_bin
        , [GlCode] Varchar(50)            collate latin1_general_bin
        , [GlCodeStart] Varchar(10)        collate latin1_general_bin
        , [GlCodeMiddle] Varchar(10)    collate latin1_general_bin
        , [GlCodeEnd] Varchar(10)        collate latin1_general_bin
        , [GlYear] Int
        , [Period1] Float
        , [Period2] Float
        , [Period3] Float
        , [Period4] Float
        , [Period5] Float
        , [Period6] Float
        , [Period7] Float
        , [Period8] Float
        , [Period9] Float
        , [Period10] Float
        , [Period11] Float
        , [Period12] Float
        --, Period13 FLOAT

        , [BeginYearBalance] Float
        , [YearMovement] Float
        );

    Insert  [#Actuals]
            ( [Company]
            , [GlCode]
            , [GlCodeStart]
            , [GlCodeMiddle]
            , [GlCodeEnd]
            , [GlYear]
            , [Period1]
            , [Period2]
            , [Period3]
            , [Period4]
            , [Period5]
            , [Period6]
```

Author: Johnson, Chris

```
            , [Period7]
            , [Period8]
            , [Period9]
            , [Period10]
            , [Period11]
            , [Period12]
            , [BeginYearBalance]
            , [YearMovement]
            )
            Select  [GH].[Company]
                  , [GH].[GlCode]
                  , [GlCodeStart] = ParseName([GH].[GlCode] , 3)
                  , [GlCodeMiddle] = ParseName([GH].[GlCode] , 2)
                  , [GlCodeEnd] = ParseName([GH].[GlCode] , 1)
                  , [GH].[GlYear]
                  , [Period1] = [GH].[ClosingBalPer1] - [GH].[BeginYearBalance]
                  , [Period2] = [GH].[ClosingBalPer2] - [GH].[ClosingBalPer1]
                  , [Period3] = [GH].[ClosingBalPer3] - [GH].[ClosingBalPer2]
                  , [Period4] = [GH].[ClosingBalPer4] - [GH].[ClosingBalPer3]
                  , [Period5] = [GH].[ClosingBalPer5] - [GH].[ClosingBalPer4]
                  , [Period6] = [GH].[ClosingBalPer6] - [GH].[ClosingBalPer5]
                  , [Period7] = [GH].[ClosingBalPer7] - [GH].[ClosingBalPer6]
                  , [Period8] = [GH].[ClosingBalPer8] - [GH].[ClosingBalPer7]
                  , [Period9] = [GH].[ClosingBalPer9] - [GH].[ClosingBalPer8]
                  , [Period10] = [GH].[ClosingBalPer10] - [GH].[ClosingBalPer9]
                  , [Period11] = [GH].[ClosingBalPer11] - [GH].[ClosingBalPer10]
                  , [Period12] = [GH].[ClosingBalPer12] - [GH].[ClosingBalPer11]
                  , [GH].[BeginYearBalance]
                  , [YearMovement] = [GH].[ClosingBalPer13] - [GH].[BeginYear-
Balance]
            From    [SysproCompany40].[dbo].[GenHistory] [GH];


--Generate Monthly Actual Amounts by unpivoting data

    Create Table [#MonthlyAmounts]
        (
          [Company] Varchar(10)                    Collate Latin1_General_BIN
        , [GlCode] Varchar(100)                    Collate Latin1_General_BIN
        , [GlCodeStart] Varchar(10)                Collate Latin1_General_BIN
        , [GlCodeMiddle] Varchar(100)             Collate Latin1_General_BIN
        , [GlCodeEnd] Varchar(10)                 Collate Latin1_General_BIN
        , [GlDeptCode] As Left([GlCodeEnd] , 1) Collate Latin1_General_BIN--
VARCHAR(10) Collate Latin1_General_BIN

        , [GlYear] Int
        , [Period] As Cast(Replace([BudgetPeriod] , 'Period' , '') As Int)
        , [BudgetPeriod] Varchar(15)             collate latin1_general_bin
        , [Movement] Float
        , [YearMovement] Float
        );

    Insert  [#MonthlyAmounts]
            ( [Company]
            , [GlCode]
            , [GlCodeStart]
            , [GlCodeMiddle]
```

Author: Johnson, Chris

```
            , [GlCodeEnd]
            --, GlDeptCode

            , [GlYear]
            , [BudgetPeriod]
            , [Movement]
            , [YearMovement]
            )
            Select  [Company]
                  , [GlCode]
                  , [GlCodeStart]
                  , [GlCodeMiddle]
                  , [GlCodeEnd]
              --, [GlDeptCode] = LEFT(GlCodeEnd, 1)

                  , [GlYear]
                  , [BudgetPeriod]
                  , [Movement] = [Actual]
                  , [YearMovement]
            From    [#Actuals] Unpivot ( [Actual] For [BudgetPeriod] In ( [Period1]
,
                                                                  [Period2] ,
                                                                  [Period3] ,
                                                                  [Period4] ,
                                                                  [Period5] ,
                                                                  [Period6] ,
                                                                  [Period7] ,
                                                                  [Period8] ,
                                                                  [Period9] ,
                                                                  [Period10] ,
                                                                  [Period11] ,
                                                                  [Period12] ) ) As
[ASMT]
    Option  ( Recompile );

--define result set

    Create Table [#Results]
        (
          [Company] Varchar(10)                       collate latin1_general_bin
        , [GlCode] Varchar(100)                       collate latin1_general_bin
        , [GlYear] Varchar(10)                       collate latin1_general_bin
        , [Period] Int
        , [Budget] Float
        , [YTDBudget] Float
        , [YearBudget] Float
        , [Movement] Float
        , [YTDMovement] Float
        , [YearMovement] Float
        , [ReportIndex2] Varchar(500)                collate latin1_general_bin
        , [Description] Varchar(150)                collate latin1_general_bin
        , [Department] Varchar(150)                   collate latin1_general_bin
        , [DepartmentDescription] Varchar(150)        collate latin1_general_bin
        , [PreviousYear] Varchar(20)                collate latin1_general_bin
        , [ReportYear] Int
        , [GlCodeStart] Varchar(25)                   collate latin1_general_bin
        , [GlCodeMiddle] Varchar(25)                collate latin1_general_bin
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-BudgetsVsActuals

```
        , [GlCodeEnd] Varchar(25)                   collate latin1_general_bin
        );

    Insert  [#Results]
            ( [Company]
            , [GlCode]
            , [GlYear]
            , [Period]
            , [Budget]
            , [YTDBudget]
            , [YearBudget]
            , [Movement]
            , [YTDMovement]
            , [YearMovement]
            , [ReportIndex2]
            , [Description]
            , [Department]
            , [DepartmentDescription]
            , [PreviousYear]
            , [ReportYear]
            , [GlCodeStart]
            , [GlCodeMiddle]
            , [GlCodeEnd]
            )
            Select  [M].[Company]
                  , [M].[GlCode]
                  , [M].[GlYear]
                  , [Period] = Coalesce([B].[BudgetPeriod] , [M].[Period])
                  , [Budget] = Coalesce([B].[Budget] , 0)
                  , [YTDBudget] = Coalesce([B].[Budget] , 0)
                  , [YearBudget] = Coalesce([B].[YearBudget] , 0)
                  , [M].[Movement]
                  , [YTDMovement] = [M].[Movement]
                  , [M].[YearMovement]
                  , [G].[ReportIndex2]
                  , [G].[Description]
                  , [Department] = Substring([M].[GlCode] , 11 , 1)
                  , [DepartmentDescription] = ''
                  , [PreviousYear] = '???'
                  , [M].[GlYear] As [ReportYear]
                  , Cast([M].[GlCodeStart] As Int)
                  , Cast([M].[GlCodeMiddle] As Int)
                  , Cast([M].[GlCodeEnd] As Int)
            From    [#UnpivotBudget] As [B]
                    Right Join [#MonthlyAmounts] As [M]
                        On [B].[Company] = [M].[Company]
                           And [B].[GlCode] = [M].[GlCode]
                           And [B].[BudgetPeriod] = [M].[Period]
                    Inner Join [SysproCompany40].[dbo].[GenMaster] As [G]
                        On [G].[Company] = [M].[Company]
                           And [G].[GlCode] = [M].[GlCode]
            Where   IsNumeric([M].[GlCodeEnd]) = 1;


--Return result set

    Select  [Company]
```

Author: Johnson, Chris

```
          , [GlCode]
          , [GlYear]
          , [Period]
          , [Budget]
          , [YTDBudget]
          , [YearBudget]
          , [Movement]
          , [YTDMovement]
          , [YearMovement]
          , [ReportIndex2]
          , [Description]
          , [Department]
          , [DepartmentDescription]
      --, PreviousYear

          , [ReportYear]
          , [GlCodeStart]
          , [GlCodeMiddle]
          , [GlCodeEnd]
    From    [#Results];

GO
EXEC sp_addextendedproperty N'MS_Description', N'budgets vs actuals from GL',
'SCHEMA', N'Report', 'PROCEDURE', N'UspResults_BudgetsVsActuals', NULL, NULL
GO
```

Uses

[Report]

[Report].[UspResults_ClosingBalancesInterco]

MS_Description

Return closing balances related to InterCompany payments for analysis

Parameters

Name	Data Type	Max Length (Bytes)
@RedTagType	char	1
@RedTagUse	varchar(500)	500

SQL Script

```
CREATE Proc [Report].[UspResults_ClosingBalancesInterco]
    (
      @RedTagType Char(1)
    , @RedTagUse Varchar(500)
    )
As
    Begin
/*
Template designed by Chris Johnson, Prometic Group March 2016
*/

--remove nocount on to speed up query

        Set NoCount On;

--Red tag

        Declare @RedTagDB Varchar(255)= Db_Name();
        Exec [Process].[UspInsert_RedTagLogs] @StoredProcDb = 'BlackBox' ,
            @StoredProcSchema = 'Report' ,
            @StoredProcName = 'UspResults_ClosingBalancesInterco' ,
            @UsedByType = @RedTagType , @UsedByName = @RedTagUse ,
            @UsedByDb = @RedTagDB;

        Create Table [#UnpivotAmounts]
            (
              [Company] Varchar(10)            collate latin1_general_bin
            , [GlCode] Varchar(35)            collate latin1_general_bin
            , [GlYear] Int
            , [Period] Int
            , [DueToCode] Varchar(10)        collate latin1_general_bin
            , [StartYearDateForExRates] DateTime2
            , [ClosingBalance] Numeric(20 , 2)
            , [MonthOffset] As Case When [Period] >= 12 Then 12
                                    Else [Period]
                               End--Period 13,14,15 should be the same as 12

            , [DateForExRates] As Case When DateAdd(Month ,
```

Author: Johnson, Chris

```
                                                    ( Case When [Period] >= 12
                                                           Then 12
                                                           Else [Period]
                                                      End ) ,
                                                    [StartYearDateForExRates]) > Get-
Date()
                                        Then GetDate()
                                        Else DateAdd(Month ,
                                                    ( Case When [Period] >= 12
                                                           Then 12
                                                           Else [Period]
                                                      End ) ,
                                                    [StartYearDateForExRates])
                                    End
            );

        Insert  [#UnpivotAmounts]
                ( [Company]
                , [GlCode]
                , [GlYear]
                , [Period]
                , [StartYearDateForExRates]
                , [ClosingBalance]
                , [DueToCode]
                )
                Select  [CBP].[Company]
                      , [CBP].[GlCode]
                      , [GlYear]
                      , [Period] = Convert(Int , Replace([Period] ,
                                                         'ClosingBalPer' , ''))
                      , [StartYearDateForExRates] = DateFromParts(Cast([GlYear] As
Int) ,
                                                                  1 , 20)
                      , [ClosingBalance]
                      , [DueToCode] = Case When [CBP].[Company] <> '10'
                                           Then Right([CBP].[GlCode] , 2)
                                           When Right([GM].[GlGroup] , 3) = 'PLI'
                                           Then '40'
                                           Else Right([CBP].[GlCode] , 2)
                                      End
                From    [SysproCompany40].[dbo].[GenHistory] Unpivot ( [Closing-
Balance] For [Period] In ( [ClosingBalPer1] ,
                                                                  [ClosingBalPer2] ,
                                                                  [ClosingBalPer3] ,
                                                                  [ClosingBalPer4] ,
                                                                  [ClosingBalPer5] ,
                                                                  [ClosingBalPer6] ,
                                                                  [ClosingBalPer7] ,
                                                                  [ClosingBalPer8] ,
                                                                  [ClosingBalPer9] ,
                                                                  [ClosingBalPer10] ,
                                                                  [ClosingBalPer11] ,
                                                                  [ClosingBalPer12] ,
                                                                  [ClosingBalPer13] ,
                                                                  [ClosingBalPer14] ,
                                                                  [ClosingBalPer15] )
)As [CBP]
                        Left Join [SysproCompany40]..[GenMaster] As [GM] On
```

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-ClosingBalancesInterco

```
[GM].[Company] = [CBP].[Company]
                                                        And [GM].[GlCode] =
[CBP].[GlCode]
                Where   [CBP].[GlCode] Not In ( 'FORCED' , 'RETAINED' );

        Select  [UA].[Company]
              , [CN].[CompanyName]
              , [CN].[ShortName]
              , [DueTo] = Coalesce([CN2].[CompanyName] , 'Unknown')
              , [DueToShortName] = Coalesce([CN2].[ShortName] , 'Unknown')
              , [GM].[GlGroup]
              , [UA].[GlCode]
              , [GlDescription] = [GM].[Description]
              , [UA].[GlYear]
              , [UA].[Period]
              , [UA].[ClosingBalance]
              , [UA].[DateForExRates]
              , [LocalCurrency] = [CN].[Currency]
              , [MultiplyRateCAD] = Convert(Numeric(16 , 4) , [CR].[CADMultiply])
              , [RateEffectiveFrom] = [CR].[StartDateTime]
              , [CADClosingBalance] = [UA].[ClosingBalance]
                * Convert(Numeric(16 , 4) , [CR].[CADMultiply])
              , [RateNotes] = 'Date for ex rates is the start of the year, plus the
number of months for the period (for example period 1, will use the date of the 20th
Feb), when this date is in the future, the current date will be used in it''s place'
        From    [#UnpivotAmounts] As [UA]
                Left Join [Lookups].[CompanyNames] As [CN] On [CN].[Company] =
[UA].[Company]
                Left Join [Lookups].[CurrencyRates] As [CR] On [CR].[Currency] =
[CN].[Currency]
                                                        And [UA].[DateForEx-
Rates] Between [CR].[StartDateTime]
                                                        And
                                                        [CR].[EndDateTime]
                Left Join [SysproCompany40].[dbo].[GenMaster] As [GM] On
[GM].[Company] = [UA].[Company]
                                                        And [GM].[GlCode] =
[UA].[GlCode]
                Left Join [Lookups].[CompanyNames] As [CN2] On [CN2].[Company] =
[UA].[DueToCode]
        Where   Upper(Left([GM].[GlGroup] , 3)) = 'ADV'
                Or Upper(Left([GM].[GlGroup] , 5)) In ( 'LTDUE' , 'INTAR' )
                Or Upper(Left([GM].[GlGroup] , 6)) = 'INTPAY';

        Drop Table [#UnpivotAmounts];

    End;
GO
EXEC sp_addextendedproperty N'MS_Description', N'Return closing balances related to
InterCompany payments for analysis', 'SCHEMA', N'Report', 'PROCEDURE', N'UspResults_-
ClosingBalancesInterco', NULL, NULL
GO
```

Uses

[Lookups].[CompanyNames]
[Lookups].[CurrencyRates]
[Process].[UspInsert_RedTagLogs]

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-ClosingBalancesInterco

[Report]

Author: Johnson, Chris

[Report].[UspResults_CompaniesCurrencyLatestExchangeRates]

MS_Description

returns current exchange rates in use

Parameters

Name	Data Type	Max Length (Bytes)
@RedTagType	char	1
@RedTagUse	varchar(500)	500

SQL Script

```
CREATE Proc [Report].[UspResults_CompaniesCurrencyLatestExchangeRates]
    (
      @RedTagType Char(1)
    , @RedTagUse Varchar(500)
    )
As /*
Template designed by Chris Johnson, Prometic Group Feb 2015
Stored procedure set out to query all live db's and return details of general ledger journal
*/
    Begin
        Set NoCount On;

--Red tag

        Declare @RedTagDB Varchar(255)= Db_Name();
        Exec [Process].[UspInsert_RedTagLogs] @StoredProcDb = 'BlackBox' ,
            @StoredProcSchema = 'Report' ,
            @StoredProcName = 'UspResults_CompaniesCurrencyLatestExchangeRates' ,
            @UsedByType = @RedTagType , @UsedByName = @RedTagUse ,
            @UsedByDb = @RedTagDB;



        Create Table [#CompanyCurrency]
            (
              [DatabaseName] Varchar(300)    collate latin1_general_bin
            , [Currency] Varchar(10)        collate latin1_general_bin
            );
        Declare @SQL Varchar(Max)= 'Use [?];

If lower(db_name()) like ''sysprocompany%'' and lower(db_name()) not like ''%srs''
and Replace(Db_Name() , ''SysproCompany'' , '''') Not In ( ''A'' , ''B'' , ''C'' ,
''D'' ,
                                                    ''E'' , ''F'' , ''G'' , ''H''
,
                                                    ''P'' , ''Q'' , ''T'' , ''U''
,
                                                    ''V'' )
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-CompaniesCurrencyLatestExchangeRates

```
begin
If Exists (Select 1 From sys.[tables] As [T] Where [T].[name] =''TblCurrency'')
    begin
    Insert [#CompanyCurrency]
            ( [DatabaseName] , [Currency] )
    Select Db_Name()
            ,[Currency] from dbo.TblCurrency
    Where [BuyEcDeclRate]=0
    End
end';

        Exec [Process].[ExecForEachDB] @cmd = @SQL;

        Select  [CC].[DatabaseName]
              , [CN].[CompanyName]
              , [CC].[Currency]
              , [CR].[StartDateTime]
              , [CR].[CADDivision]
              , [CR].[CHFDivision]
              , [CR].[EURDivision]
              , [CR].[GBPDivision]
              , [CR].[JPYDivision]
              , [CR].[USDDivision]
              , [CR].[CADMultiply]
              , [CR].[CHFMultiply]
              , [CR].[EURMultiply]
              , [CR].[GBPMultiply]
              , [CR].[JPYMultiply]
              , [CR].[USDMultiply]
              , [CR].[LastUpdated]
        From    [#CompanyCurrency] As [CC]
                Left Join [Lookups].[CurrencyRates] As [CR]
                    On [CR].[Currency] = [CC].[Currency]
                       And GetDate() Between [CR].[StartDateTime]
                                     And     [CR].[EndDateTime]
                Left Join [Lookups].[CompanyNames] As [CN]
                    On Replace([CC].[DatabaseName] , 'SysproCompany' , '') =
[CN].[Company];

        Drop Table [#CompanyCurrency];
    End;
GO
EXEC sp_addextendedproperty N'MS_Description', N'returns current exchange rates in
use', 'SCHEMA', N'Report', 'PROCEDURE', N'UspResults_CompaniesCurrencyLatestExchange-
Rates', NULL, NULL
GO
```

Uses

[Lookups].[CompanyNames]
[Lookups].[CurrencyRates]
[Process].[ExecForEachDB]
[Process].[UspInsert_RedTagLogs]
[Report]

Author: Johnson, Chris

[Report].[UspResults_CompanyTransactions]

MS_Description

journals for each company

Parameters

Name	Data Type	Max Length (Bytes)
@Company	varchar(max)	max
@RedTagType	char	1
@RedTagUse	varchar(500)	500

SQL Script

```
CREATE Proc [Report].[UspResults_CompanyTransactions]
     (
       @Company Varchar(Max)
     , @RedTagType Char(1)
     , @RedTagUse Varchar(500)
     )
 As
     Begin
/*
Template designed by Chris Johnson, Prometic Group September 2015
Stored procedure set out to query multiple databases with the same information and
return it in a collated format
*/
        If IsNumeric(@Company) = 0
            Begin
                Select  @Company = Upper(@Company);
            End;

--remove nocount on to speed up query

        Set NoCount On;

--Red tag

        Declare @RedTagDB Varchar(255)= Db_Name();
        Exec [Process].[UspInsert_RedTagLogs] @StoredProcDb = 'BlackBox' ,
            @StoredProcSchema = 'Report' ,
            @StoredProcName = 'UspResults_CompanyTransactions' ,
            @UsedByType = @RedTagType , @UsedByName = @RedTagUse ,
            @UsedByDb = @RedTagDB;

--list the tables that are to be pulled back from each DB - if they are not found
the script will not be run against that db

        Declare @ListOfTables Varchar(Max) = 'AssetDepreciation,TblApTerms';

--create temporary tables to be pulled from different databases, including a column
```

Author: Johnson, Chris

```
to id

        Create Table [#GenJournalDetail]
            (
              [DatabaseName] Varchar(150)        collate latin1_general_bin
            , [Journal] Int
            , [GlYear] Int
            , [GlPeriod] Int
            , [EntryNumber] Int
            , [EntryType] Char(1)                collate latin1_general_bin
            , [GlCode] Varchar(35)                collate latin1_general_bin
            , [Reference] Varchar(50)            collate latin1_general_bin
            , [Comment] Varchar(250)            collate latin1_general_bin
            , [EntryValue] Numeric(20 , 2)
            , [EntryDate] Date
            , [EntryPosted] Char(1)                collate latin1_general_bin
            );
        Create Table [#GenJournalCtl]
            (
              [DatabaseName] Varchar(150)        collate latin1_general_bin
            , [JnlPrintFlag] Char(1)            collate latin1_general_bin
            , [JournalDate] Date
            , [NumOfEntries] Int
            , [DebitAmount] Numeric(20 , 2)
            , [CreditAmount] Numeric(20 , 2)
            , [JnlPostingType] Char(1)            collate latin1_general_bin
            , [Source] Char(1)                    collate latin1_general_bin
            , [Operator] Varchar(20)            collate latin1_general_bin
            , [JnlStatus] Char(1)                collate latin1_general_bin
            , [Reference] Varchar(30)            collate latin1_general_bin
            , [AuthorisedBy] Varchar(20)        collate latin1_general_bin
            , [PostedBy] Varchar(20)            collate latin1_general_bin
            , [Authorised] Char(1)                collate latin1_general_bin
            , [PostDate] Date
            , [Notation] Varchar(100)            collate latin1_general_bin
            , [GlJournal] Int
            , [GlPeriod] Int
            , [GlYear] Int
            , [JournalSource] Char(2)            collate latin1_general_bin
            );

--create script to pull data from each db into the tables

        Declare @SQLGenJournalDetail Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables
            + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
```

```
String](@ListOfTables,'','')
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'',''))
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
                    Insert [#GenJournalDetail]
                            ( [DatabaseName]
                            , [Journal]
                            , [GlYear]
                            , [GlPeriod]
                            , [EntryNumber]
                            , [EntryType]
                            , [GlCode]
                            , [Reference]
                            , [Comment]
                            , [EntryValue]
                            , [EntryDate]
                            , [EntryPosted]
                            )
                    SELECT [DatabaseName]=@DBCode
                         , [GJD].[Journal]
                         , [GJD].[GlYear]
                         , [GJD].[GlPeriod]
                         , [GJD].[EntryNumber]
                         , [GJD].[EntryType]
                         , [GJD].[GlCode]
                         , [GJD].[Reference]
                         , [GJD].[Comment]
                         , [GJD].[EntryValue]
                         , [GJD].[EntryDate]
                         , [GJD].[EntryPosted] FROM [GenJournalDetail] As [GJD]
            End
    End';
        Declare @SQLGenJournalCtl Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end'
            + --Only query DBs beginning SysProCompany

            '
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN'
            + --only companies selected in main run, or if companies selected then
all

            '
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT'
            + --count number of tables requested (number of commas plus one)

            '
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')'
```

Author: Johnson, Chris

```
            + --Count of the tables requested how many exist in the db

            '
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'',')) '
            + --only if the count matches (all the tables exist in the requested db)
then run the script

            '
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
                    Insert [#GenJournalCtl]
                            ( [DatabaseName]
                            , [JnlPrintFlag]
                            , [JournalDate]
                            , [NumOfEntries]
                            , [DebitAmount]
                            , [CreditAmount]
                            , [JnlPostingType]
                            , [Source]
                            , [Operator]
                            , [JnlStatus]
                            , [Reference]
                            , [AuthorisedBy]
                            , [PostedBy]
                            , [Authorised]
                            , [PostDate]
                            , [Notation]
                            , [GlJournal]
                            , [GlPeriod]
                            , [GlYear]
                            , [JournalSource]
                            )
                    SELECT [DatabaseName]=@DBCode
                         , [GJC].[JnlPrintFlag]
                         , [GJC].[JournalDate]
                         , [GJC].[NumOfEntries]
                         , [GJC].[DebitAmount]
                         , [GJC].[CreditAmount]
                         , [GJC].[JnlPostingType]
                         , [GJC].[Source]
                         , [GJC].[Operator]
                         , [GJC].[JnlStatus]
                         , [GJC].[Reference]
                         , [GJC].[AuthorisedBy]
                         , [GJC].[PostedBy]
                         , [GJC].[Authorised]
                         , [GJC].[PostDate]
                         , [GJC].[Notation]
                         , [GJC].[GlJournal]
                         , [GJC].[GlPeriod]
                         , [GJC].[GlYear]
                         , [GJC].[JournalSource] FROM [GenJournalCtl] As [GJC]
            End
    End';
--Enable this function to check script changes (try to run script directly against
db manually)
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-CompanyTransactions

```
--Print @SQLGenJournalCtl

--Print @SQLGenJournalDetail


--execute script against each db, populating the base tables

        Exec [Process].[ExecForEachDB] @cmd = @SQLGenJournalCtl;
        Exec [Process].[ExecForEachDB] @cmd = @SQLGenJournalDetail;

--define the results you want to return

        Create Table [#Results]
            (
              [DatabaseName] Varchar(150)        collate latin1_general_bin
            , [Journal] Int
            , [GlYear] Int
            , [GlPeriod] Int
            , [EntryNumber] Int
            , [EntryType] Char(1)                collate latin1_general_bin
            , [GlCode] Varchar(35)                collate latin1_general_bin
            , [Reference] Varchar(50)            collate latin1_general_bin
            , [Comment] Varchar(250)            collate latin1_general_bin
            , [EntryValue] Numeric(20 , 2)
            , [EntryDate] Date
            , [EntryPosted] Char(1)                collate latin1_general_bin
            , [JnlPrintFlag] Char(1)            collate latin1_general_bin
            , [JournalDate] Date
            , [NumOfEntries] Int
            , [DebitAmount] Numeric(20 , 2)
            , [CreditAmount] Numeric(20 , 2)
            , [JnlPostingType] Char(1)            collate latin1_general_bin
            , [Source] Char(1)                    collate latin1_general_bin
            , [Operator] Varchar(20)            collate latin1_general_bin
            , [JnlStatus] Char(1)                collate latin1_general_bin
            , [AuthorisedBy] Varchar(20)        collate latin1_general_bin
            , [PostedBy] Varchar(20)            collate latin1_general_bin
            , [Authorised] Char(1)                collate latin1_general_bin
            , [PostDate] Date
            , [Notation] Varchar(100)            collate latin1_general_bin
            , [JournalSource] Char(2)            collate latin1_general_bin
            );

--Placeholder to create indexes as required


--script to combine base data and insert into results table

        Insert  [#Results]
                ( [DatabaseName]
                , [Journal]
                , [GlYear]
                , [GlPeriod]
                , [EntryNumber]
                , [EntryType]
                , [GlCode]
```

Author: Johnson, Chris

```
                , [Reference]
                , [Comment]
                , [EntryValue]
                , [EntryDate]
                , [EntryPosted]
                , [JnlPrintFlag]
                , [JournalDate]
                , [NumOfEntries]
                , [DebitAmount]
                , [CreditAmount]
                , [JnlPostingType]
                , [Source]
                , [Operator]
                , [JnlStatus]
                , [AuthorisedBy]
                , [PostedBy]
                , [Authorised]
                , [PostDate]
                , [Notation]
                , [JournalSource]
                )
                Select  [GJD].[DatabaseName]
                      , [GJD].[Journal]
                      , [GJD].[GlYear]
                      , [GJD].[GlPeriod]
                      , [GJD].[EntryNumber]
                      , [GJD].[EntryType]
                      , [GJD].[GlCode]
                      , [GJD].[Reference]
                      , [GJD].[Comment]
                      , [GJD].[EntryValue]
                      , [GJD].[EntryDate]
                      , [GJD].[EntryPosted]
                      , [GJC].[JnlPrintFlag]
                      , [GJC].[JournalDate]
                      , [GJC].[NumOfEntries]
                      , [GJC].[DebitAmount]
                      , [GJC].[CreditAmount]
                      , [GJC].[JnlPostingType]
                      , [GJC].[Source]
                      , [GJC].[Operator]
                      , [GJC].[JnlStatus]
                      , [GJC].[AuthorisedBy]
                      , [GJC].[PostedBy]
                      , [GJC].[Authorised]
                      , [GJC].[PostDate]
                      , [GJC].[Notation]
                      , [GJC].[JournalSource]
                From    [#GenJournalDetail] As [GJD]
                        Left Join [#GenJournalCtl] As [GJC]
                            On [GJC].[GlJournal] = [GJD].[Journal]
                               And [GJC].[GlPeriod] = [GJD].[GlPeriod]
                               And [GJC].[GlYear] = [GJD].[GlYear]
                               And [GJC].[DatabaseName] = [GJD].[DatabaseName];

--return results
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-CompanyTransactions

```
        Select  [R].[DatabaseName]
              , [CN].[CompanyName]
              , [R].[Journal]
              , [R].[GlYear]
              , [R].[GlPeriod]
              , [R].[EntryNumber]
              , [R].[EntryType]
              , [R].[GlCode]
              , [GLDescription] = [GM].[Description]
              , [R].[Reference]
              , [R].[Comment]
              , [R].[EntryValue]
              , [R].[EntryDate]
              , [R].[EntryPosted]
              , [R].[JnlPrintFlag]
              , [R].[JournalDate]
              , [R].[NumOfEntries]
              , [R].[DebitAmount]
              , [R].[CreditAmount]
              , [JPT].[JnlPostingTypeDesc]
              , [GJCS].[SourceDescription]
              , [R].[Operator]
              , [JS].[JnlStatusDesc]
              , [R].[AuthorisedBy]
              , [R].[PostedBy]
              , [R].[Authorised]
              , [R].[PostDate]
              , [R].[Notation]
              , [JournalSource] = [GJCJS].[GenJournalCtlJnlSourceDesc]
        From    [#Results] As [R]
                Left Join [BlackBox].[Lookups].[JnlPostingType] [JPT]
                    On [R].[JnlPostingType] = [JPT].[JnlPostingType]
                Left Join [BlackBox].[Lookups].[JnlStatus] [JS]
                    On [R].[JnlStatus] = [JS].[JnlStatus]
                Left Join [BlackBox].[Lookups].[GenJournalCtlSource] [GJCS]
                    On [R].[Source] = [GJCS].[Source]
                Left Join [SysproCompany40].[dbo].[GenMaster] As [GM]
                    On [GM].[GlCode] = [R].[GlCode]
                       And [GM].[Company] = [R].[DatabaseName]
                Left Join [BlackBox].[Lookups].[CompanyNames] As [CN]
                    On [CN].[Company] = [R].[DatabaseName]
                Left Join [Lookups].[GenJournalCtlJnlSource] [GJCJS]
                    On [R].[JournalSource] = [GJCJS].[GenJournalCtlJnlSource];

    End;

GO
EXEC sp_addextendedproperty N'MS_Description', N'journals for each company',
'SCHEMA', N'Report', 'PROCEDURE', N'UspResults_CompanyTransactions', NULL, NULL
GO
```

Uses

[Lookups].[CompanyNames]
[Lookups].[GenJournalCtlJnlSource]

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-CompanyTransactions

[Lookups].[GenJournalCtlSource]

[Lookups].[JnlPostingType]

[Lookups].[JnlStatus]

[Process].[ExecForEachDB]

[Process].[UspInsert_RedTagLogs]

[Report]

Author: Johnson, Chris

[Report].[UspResults_ExecForEachDBLogs]

MS_Description

returns details of how often exec for each db is being called, used to monitor code usage

SQL Script

```
Create Proc [Report].[UspResults_ExecForEachDBLogs]
as
--get total number of logs for % calc

Declare @LogTotal Numeric(5,2)

Select @LogTotal=Count(Distinct [EFEDL].[LogID])
From [History].[ExecForEachDBLogs] As [EFEDL]

--Return results

Select  [LogDate] = Cast([EFEDL].[LogTime] As Date)
      , [DayOfTheWeek] = DateName(Weekday , [EFEDL].[LogTime])
      , [CountOfLogs] = Count(Distinct [EFEDL].[LogID])
      , [CountOfLogsPercent] = (Count(Distinct [EFEDL].[LogID])/@LogTotal)*100
From    [History].[ExecForEachDBLogs] As [EFEDL]
Group By Cast([EFEDL].[LogTime] As Date)
      , DateName(Weekday , [EFEDL].[LogTime])
Order By [LogDate] Asc;

GO
EXEC sp_addextendedproperty N'MS_Description', N'returns details of how often exec
for each db is being called, used to monitor code usage', 'SCHEMA', N'Report',
'PROCEDURE', N'UspResults_ExecForEachDBLogs', NULL, NULL
GO
```

Uses

[History].[ExecForEachDBLogs]
[Report]

Author: Johnson, Chris

[Report].[UspResults_FixedAssets]

MS_Description

details of all fixed assets

Parameters

Name	Data Type	Max Length (Bytes)
@Company	varchar(max)	max
@RedTagType	char	1
@RedTagUse	varchar(500)	500

SQL Script

```
CREATE Proc [Report].[UspResults_FixedAssets]
    (
      @Company Varchar(Max)
    , @RedTagType Char(1)
    , @RedTagUse Varchar(500)
    )
As /*
Template designed by Chris Johnson, Prometic Group September 2015
Stored procedure set out to query multiple databases with the same information and
return it in a collated format
Exec [Report].[UspResults_FixedAssets] 10
*/
    Begin

        Set NoCount On;

        If IsNumeric(@Company) = 0
            Begin
                Select  @Company = Upper(@Company);
            End;



--Red tag

        Declare @RedTagDB Varchar(255)= Db_Name();
        Exec [Process].[UspInsert_RedTagLogs] @StoredProcDb = 'BlackBox' ,
            @StoredProcSchema = 'Report' ,
            @StoredProcName = 'UspResults_FixedAssets' ,
            @UsedByType = @RedTagType , @UsedByName = @RedTagUse ,
            @UsedByDb = @RedTagDB;

--list the tables that are to be pulled back from each DB - if they are not found
the script will not be run against that db

        Declare @ListOfTables Varchar(Max) = 'AssetMaster,AssetType,AssetLocation';
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-FixedAssets

```
--create temporary tables to be pulled from different databases, including a column
to id

        Create Table [#AssetMaster]
            (
              [DatabaseName] Varchar(150) Collate Latin1_General_BIN
            , [Asset] Varchar(50) Collate Latin1_General_BIN
            , [Description] Varchar(100) Collate Latin1_General_BIN
            , [AssetQty] Numeric(20 , 7)
            , [OriginalAssetQty] Numeric(20 , 7)
            , [OriginalAssetValue] Numeric(20 , 7)
            , [GlCode] Varchar(150) Collate Latin1_General_BIN
            , [AssetType] Varchar(50) Collate Latin1_General_BIN
            , [Location] Varchar(50) Collate Latin1_General_BIN
            , [AssetGroupCode] Varchar(10) Collate Latin1_General_BIN
            , [PurchaseDate] Date
            , [FirstInstalDate] Date
            , [DateSold] Date
            , [DisposedFlag] Char(1) Collate Latin1_General_BIN
            , [DisposalReason] Varchar(10) Collate Latin1_General_BIN
            );
        Create Table [#AssetType]
            (
              [DatabaseName] Varchar(150) Collate Latin1_General_BIN
            , [AssetType] Varchar(50) Collate Latin1_General_BIN
            , [Description] Varchar(100) Collate Latin1_General_BIN
            );
        Create Table [#AssetLocation]
            (
              [DatabaseName] Varchar(150) Collate Latin1_General_BIN
            , [Location] Varchar(50) Collate Latin1_General_BIN
            , [Description] Varchar(100) Collate Latin1_General_BIN
            );
        Create Table [#AssetGroup]
            (
              [DatabaseName] Varchar(150) Collate Latin1_General_BIN
            , [AssetGroupCode] Varchar(10) Collate Latin1_General_BIN
            , [Description] Varchar(100) Collate Latin1_General_BIN
            );
        Create Table [#AssetReasonDisp]
            (
              [DatabaseName] Varchar(150) Collate Latin1_General_BIN
            , [DisposalReason] Varchar(10) Collate Latin1_General_BIN
            , [Description] Varchar(50) Collate Latin1_General_BIN
            );

--create script to pull data from each db into the tables

        Declare @SQL1 Varchar(Max) = '    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            BEGIN
```

Author: Johnson, Chris

```
                Insert [#AssetMaster]
                ( [DatabaseName]
                , [Asset]
                , [Description]
                , [AssetQty]
                , [OriginalAssetQty]
                , [OriginalAssetValue]
                , [GlCode]
                , [AssetType]
                , [Location]
                , [AssetGroupCode]
                , [PurchaseDate]
                , [FirstInstalDate]
                , [DateSold]
                , [DisposedFlag]
                , [DisposalReason]
                )
                SELECT [DatabaseName] = @DBCode
                , [am].[Asset]
                , [am].[Description]
                , [am].[AssetQty]
                , [am].[OriginalAssetQty]
                , [am].[OriginalAssetValue]
                , [am].[GlCode]
                , [am].[AssetType]
                , [am].[Location]
                , [am].[AssetGroupCode]
                , [am].PurchaseDate
                , [am].FirstInstalDate
                , [am].DateSold
                , [am].[DisposedFlag]
                , [am].[DisposalReason]
                From [AssetMaster] As [am]
            End
    End';
        Declare @SQL2 Varchar(Max) = 'USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            BEGIN
                Insert [#AssetType]
                    ( [DatabaseName]
                    , [AssetType]
                    , [Description]
                    )
                SELECT [DatabaseName]=@DBCode
                     , [at].[AssetType]
                     , [at].[Description]
                FROM [AssetType] As [at]
            End
    End';
        Declare @SQL3 Varchar(Max) = 'USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-FixedAssets

```
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            BEGIN
                Insert [#AssetLocation]
                ( [DatabaseName]
                , [Location]
                , [Description]
                )
                SELECT [DatabaseName]=@DBCode
                     , [al].[Location]
                     , [al].[Description]
                FROM [AssetLocation] As [al]
            End
    End';
        Declare @SQL4 Varchar(Max) = 'USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            BEGIN
            Insert [#AssetGroup]
            ( [DatabaseName]
            , [AssetGroupCode]
            , [Description]
            )
            SELECT   @DBCode
                    ,[ag].[AssetGroupCode]
                    ,[ag].[Description]
             FROM [dbo].[AssetGroup] As [ag]
            End
    End';
        Declare @SQL5 Varchar(Max) = 'USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            BEGIN
            Insert  [#AssetReasonDisp]
                ( [DatabaseName]
                , [DisposalReason]
                , [Description]
                )
                Select  @DBCode
                      , [ARD].[DisposalReason]
                      , [ARD].[Description]
                From    [AssetReasonDisp] [ARD];
            End
    End';
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-FixedAssets

```
--Enable this function to check script changes (try to run script directly against
db manually)

--Print @SQL


--execute script against each db, populating the base tables

        Exec [Process].[ExecForEachDB_WithTableCheck] @cmd = @SQL1 ,
            @SchemaTablesToCheck = @ListOfTables;
        Exec [Process].[ExecForEachDB_WithTableCheck] @cmd = @SQL2 ,
            @SchemaTablesToCheck = @ListOfTables;
        Exec [Process].[ExecForEachDB_WithTableCheck] @cmd = @SQL3 ,
            @SchemaTablesToCheck = @ListOfTables;
        Exec [Process].[ExecForEachDB_WithTableCheck] @cmd = @SQL4 ,
            @SchemaTablesToCheck = @ListOfTables;
        Exec [Process].[ExecForEachDB_WithTableCheck] @cmd = @SQL5 ,
            @SchemaTablesToCheck = @ListOfTables;

--define the results you want to return

        Create Table [#Results]
            (
              [DatabaseName] Varchar(150) Collate Latin1_General_BIN
            , [Asset] Varchar(50) Collate Latin1_General_BIN
            , [Description] Varchar(100) Collate Latin1_General_BIN
            , [Location] Varchar(100) Collate Latin1_General_BIN
            , [AssetType] Varchar(100) Collate Latin1_General_BIN
            , [AssetQty] Numeric(20 , 7)
            , [OriginalAssetQty] Numeric(20 , 7)
            , [OriginalAssetValue] Numeric(20 , 7)
            , [GlCode] Varchar(150) Collate Latin1_General_BIN
            , [AssetGroup] Varchar(150) Collate Latin1_General_BIN
            , [PurchaseDate] Date
            , [FirstInstalDate] Date
            , [DateSold] Date
            , [DisposedFlag] Char(1) Collate Latin1_General_BIN
            , [DisposalReason] Varchar(50) Collate Latin1_General_BIN
            );

--Placeholder to create indexes as required

--create NonClustered Index Index_Name On #Table1 (DatabaseName) Include (Column-
Name)


--script to combine base data and insert into results table

        Insert  [#Results]
                ( [DatabaseName]
                , [Asset]
                , [Description]
                , [Location]
                , [AssetType]
                , [AssetQty]
                , [OriginalAssetQty]
                , [OriginalAssetValue]
```

Author: Johnson, Chris

```
                , [GlCode]
                , [AssetGroup]
                , [PurchaseDate]
                , [FirstInstalDate]
                , [DateSold]
                , [DisposedFlag]
                , [DisposalReason]
                )
                Select  [am].[DatabaseName]
                      , [am].[Asset]
                      , [am].[Description]
                      , [Location] = [al].[Description]
                      , [AssetType] = [at].[Description]
                      , [am].[AssetQty]
                      , [am].[OriginalAssetQty]
                      , [am].[OriginalAssetValue]
                      , [am].[GlCode]
                      , [ag].[Description]
                      , [am].[PurchaseDate]
                      , [am].[FirstInstalDate]
                      , [am].[DateSold]
                      , [DisposedFlag] = Case When [am].[DisposedFlag] = ''
                                              Then Null
                                              Else [am].[DisposedFlag]
                                         End
                      , [ARD].[Description]
                From    [#AssetMaster] As [am]
                        Left Join [#AssetType] As [at]
                            On [at].[AssetType] = [am].[AssetType]
                               And [at].[DatabaseName] = [am].[DatabaseName]
                        Left Join [#AssetLocation] As [al]
                            On [al].[Location] = [am].[Location]
                               And [al].[DatabaseName] = [am].[DatabaseName]
                        Left Join [#AssetGroup] As [ag]
                            On [ag].[AssetGroupCode] = [am].[AssetGroupCode]
                               And [ag].[DatabaseName] = [am].[DatabaseName]
                        Left Join [#AssetReasonDisp] [ARD]
                            On [ARD].[DisposalReason] = [am].[DisposalReason];

        Set NoCount Off;
--return results

        Select  [cn].[CompanyName]
              , [r].[Asset]
              , [r].[Description]
              , [r].[Location]
              , [r].[AssetType]
              , [r].[AssetQty]
              , [r].[OriginalAssetQty]
              , [r].[OriginalAssetValue]
              , [r].[GlCode]
              , [r].[AssetGroup]
              , [r].[PurchaseDate]
              , [r].[FirstInstalDate]
              , [r].[DateSold]
              , [r].[DisposedFlag]
              , [r].[DisposalReason]
```

Author: Johnson, Chris

```
        From    [#Results] [r]
                Left Join [Lookups].[CompanyNames] As [cn]
                    On [r].[DatabaseName] = [cn].[Company];

    End;

GO
EXEC sp_addextendedproperty N'MS_Description', N'details of all fixed assets',
'SCHEMA', N'Report', 'PROCEDURE', N'UspResults_FixedAssets', NULL, NULL
GO
```

Uses

[Lookups].[CompanyNames]
[Process].[ExecForEachDB_WithTableCheck]
[Process].[UspInsert_RedTagLogs]
[Report]

[Report].[UspResults_GeneralLedgerControlStats]

MS_Description

details from Gen Control tables, providing quick overview of ledger

Parameters

Name	Data Type	Max Length (Bytes)
@RedTagType	char	1
@RedTagUse	varchar(500)	500

SQL Script

```
Create Proc [Report].[UspResults_GeneralLedgerControlStats]
    (
      @RedTagType Char(1)
    , @RedTagUse Varchar(500)
    )
As
    Begin
/*
Template designed by Chris Johnson, Prometic Group April 2016
*/

--Red tag

        Declare @RedTagDB Varchar(255)= Db_Name();
        Exec [Process].[UspInsert_RedTagLogs] @StoredProcDb = 'BlackBox' ,
            @StoredProcSchema = 'Report' ,
            @StoredProcName = 'UspResults_GeneralLedgerControlStats' ,
            @UsedByType = @RedTagType , @UsedByName = @RedTagUse ,
            @UsedByDb = @RedTagDB;

        Select  [CN].[CompanyName]
              , [CN].[ShortName]
              , [CN].[Currency]
              , [GC].[Company]
              , [GC].[GlYear]
              , [GC].[GlPeriod]
              , [GC].[CurBalAsset]
              , [GC].[CurBalLiability]
              , [GC].[CurBalCapital]
              , [GC].[CurBalRevenue]
              , [GC].[CurBalExpense]
              , [GC].[OldestYearDet]
              , [GC].[CurYrPrdEnd1]
              , [GC].[CurYrPrdEnd2]
              , [GC].[CurYrPrdEnd3]
              , [GC].[CurYrPrdEnd4]
              , [GC].[CurYrPrdEnd5]
              , [GC].[CurYrPrdEnd6]
```

Author: Johnson, Chris

```
                , [GC].[CurYrPrdEnd7]
                , [GC].[CurYrPrdEnd8]
                , [GC].[CurYrPrdEnd9]
                , [GC].[CurYrPrdEnd10]
                , [GC].[CurYrPrdEnd11]
                , [GC].[CurYrPrdEnd12]
                , [GC].[CurYrPrdEnd13]
                , [GC].[CurYrPrdEnd14]
                , [GC].[CurYrPrdEnd15]
                , [GC].[PrvYrPrdEnd1]
                , [GC].[PrvYrPrdEnd2]
                , [GC].[PrvYrPrdEnd3]
                , [GC].[PrvYrPrdEnd4]
                , [GC].[PrvYrPrdEnd5]
                , [GC].[PrvYrPrdEnd6]
                , [GC].[PrvYrPrdEnd7]
                , [GC].[PrvYrPrdEnd8]
                , [GC].[PrvYrPrdEnd9]
                , [GC].[PrvYrPrdEnd10]
                , [GC].[PrvYrPrdEnd11]
                , [GC].[PrvYrPrdEnd12]
                , [GC].[PrvYrPrdEnd13]
                , [GC].[PrvYrPrdEnd14]
                , [GC].[PrvYrPrdEnd15]
                , [GC].[AuthoriseJournals]
        From    [SysproCompany40].[dbo].[GenControl] [GC]
                Left Join [BlackBox].[Lookups].[CompanyNames] [CN]
                    On [CN].[Company] = [GC].[Company];
    End;

GO
EXEC sp_addextendedproperty N'MS_Description', N'details from Gen Control tables,
providing quick overview of ledger', 'SCHEMA', N'Report', 'PROCEDURE', N'UspResults_-
GeneralLedgerControlStats', NULL, NULL
GO
```

Uses

[Lookups].[CompanyNames]
[Process].[UspInsert_RedTagLogs]
[Report]

Author: Johnson, Chris

[Report].[UspResults_GenJournalDetails]

MS_Description

list of journals

Parameters

Name	Data Type	Max Length (Bytes)
@RedTagType	char	1
@RedTagUse	varchar(500)	500

SQL Script

```
CREATE Proc [Report].[UspResults_GenJournalDetails]
    (
      @RedTagType Char(1)
    , @RedTagUse Varchar(500)
    )
As /*
Template designed by Chris Johnson, Prometic Group Feb 2015
Stored procedure set out to query all live db's and return details of general ledger
journal
*/
    Set NoCount On;

--Red tag

    Declare @RedTagDB Varchar(255)= Db_Name()
      , @Company Varchar(255)= 'ALL';
    Exec [Process].[UspInsert_RedTagLogs] @StoredProcDb = 'BlackBox' ,
        @StoredProcSchema = 'Report' ,
        @StoredProcName = 'UspResults_GenJournalDetails' ,
        @UsedByType = @RedTagType , @UsedByName = @RedTagUse ,
        @UsedByDb = @RedTagDB;

--list the tables that are to be pulled back from each DB - if they are not found
the script will not be run against that db

    Declare @ListOfTables Varchar(Max) = 'GenJournalDetail,GrnMatching,InvJournal-
Det';

    Create Table [#GenJournalDetail]
        (
          [DatabaseName] Varchar(300)        collate latin1_general_bin
        , [SourceDetail] Varchar(100)        collate latin1_general_bin
        , [GlYear] Int
        , [GlPeriod] Int
        , [Journal] Int
        , [EntryNumber] Int
        , [EntryType] Char(1)                collate latin1_general_bin
        , [GlCode] Varchar(35)                collate latin1_general_bin
```

Author: Johnson, Chris

```
        , [Reference] Varchar(100)            collate latin1_general_bin
        , [Comment] Varchar(250)            collate latin1_general_bin
        , [EntryValue] Numeric(20 , 2)
        , [InterCompanyFlag] Char(1)        collate latin1_general_bin
        , [Company] Varchar(50)               collate latin1_general_bin
        , [EntryDate] Date
        , [EntryPosted] Char(1)               collate latin1_general_bin
        , [CurrencyValue] Numeric(20 , 2)
        , [PostCurrency] Varchar(10)        collate latin1_general_bin
        , [TypeDetail] Varchar(100)           collate latin1_general_bin
        , [CommitmentFlag] Char(1)           collate latin1_general_bin
        , [TransactionDate] Date
        , [DocumentDate] Date
        , [SubModJournal] Int
        , [AnalysisEntry] Int
        );
    Create Table [#GrnMatching]
        (
          [DatabaseName] Varchar(300)        collate latin1_general_bin
        , [Grn] Varchar(20)                   collate latin1_general_bin
        , [Invoice] Varchar(20)               collate latin1_general_bin
        );
    Create Table [#InvJournalDet]
        (
          [DatabaseName] Varchar(300)        collate latin1_general_bin
        , [JnlYear] Int
        , [GlPeriod] Int
        , [Journal] Int
        , [EntryNumber] Int
        , [Supplier] Varchar(15)            collate latin1_general_bin
        , [PurchaseOrder] Varchar(20)        collate latin1_general_bin
        , [Reference] Varchar(30)            collate latin1_general_bin
        );
    Create Table [#GenAnalysisTrn]
        (
          [DatabaseName] Varchar(150)        collate latin1_general_bin
        , [AnalysisEntry] Int
        , [GlPeriod] Int
        , [GlYear] Int
        , [AnalysisCategory] Varchar(10)    collate latin1_general_bin
        , [AnalysisCode1] Varchar(10)        collate latin1_general_bin
        , [AnalysisCode2] Varchar(10)        collate latin1_general_bin
        , [AnalysisCode3] Varchar(10)        collate latin1_general_bin
        , [AnalysisCode4] Varchar(10)        collate latin1_general_bin
        , [AnalysisCode5] Varchar(10)        collate latin1_general_bin
        );
    Create Table [#GenAnalysisCode]
        (
          [DatabaseName] Varchar(150)        collate latin1_general_bin
        , [AnalysisCategory] Varchar(10)    collate latin1_general_bin
        , [AnalysisCode] Varchar(10)        collate latin1_general_bin
        , [AnalysisType] Int
        , [Description] Varchar(50)           collate latin1_general_bin
        );

    Declare @SQLGenJournalDetail Varchar(Max) = '
USE [?];
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_Gen-JournalDetails

```
Declare @DB varchar(150),@DBCode varchar(150)
Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end
IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS'' and Is-
Numeric(Replace(Db_Name(),''SysproCompany'',''''))=1
BEGIN
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
        + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables
        + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'',''))
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
            Declare @SubSQL Varchar(max)

            Select @SubSQL=''SELECT  SourceDetail = Coalesce([GJDS].[GJSourceDetail]
,
                                ''''No Source'''')
            , [GJD].[GlYear]
            , [GJD].[GlPeriod]
            , [GJD].[Journal]
            , [GJD].[EntryNumber]
            , [GJD].[EntryType]
            , [GJD].[GlCode]
            , [GJD].[Reference]
            , [GJD].[Comment]
            , [GJD].[EntryValue]
            , [GJD].[InterCompanyFlag]
            , [Company] = case when [GJD].[Company]=''''''''  then
''''''+@DBCode+'''''' else [GJD].[Company] end
            , [GJD].[EntryDate]
            , [GJD].[EntryPosted]
            , [GJD].[CurrencyValue]
            , [GJD].[PostCurrency]
            , [TypeDetail] = Coalesce([GJT].[TypeDetail] ,
                                ''''No Type'''')
            , [GJD].[CommitmentFlag]
            , [GJD].[TransactionDate]
            , [GJD].[DocumentDate]
            , DatabaseName = ''''''+@DBCode+''''''
            , [GJD].[SubModJournal]
            , [GJD].[AnalysisEntry]
        From    [dbo].[GenJournalDetail] As [GJD]
            Left Join [BlackBox].[Lookups].[GenJournalDetailSource]
                As [GJDS]
                On [GJDS].[GJSource] = [GJD].[Source]
            Left Join [BlackBox].[Lookups].[GenJournalType]
                As [GJT]
                On [GJD].[Type] = [GJT].[TypeCode]
                    And [GJD].[SubModWh] <> ''''RM'''';''

        Insert  [#GenJournalDetail]
```

Author: Johnson, Chris

```
            ( [SourceDetail]
            , [GlYear]
            , [GlPeriod]
            , [Journal]
            , [EntryNumber]
            , [EntryType]
            , [GlCode]
            , [Reference]
            , [Comment]
            , [EntryValue]
            , [InterCompanyFlag]
            , [Company]
            , [EntryDate]
            , [EntryPosted]
            , [CurrencyValue]
            , [PostCurrency]
            , [TypeDetail]
            , [CommitmentFlag]
            , [TransactionDate]
            , [DocumentDate]
            , [DatabaseName]
            , [SubModJournal]
            , [AnalysisEntry]
            )
        Exec (@SubSQL)
    End
End';
    Declare @SQLGrnMatching Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS'' and Is-
Numeric(Replace(Db_Name(),''SysproCompany'',''''))=1
    BEGIN
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
        + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables
        + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'',''))
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
                Insert  [#GrnMatching]
                        ( [DatabaseName]
                        , [Grn]
                        , [Invoice]
                        )
                        SELECT  [DatabaseName] = @DBCode
                              , [GM].[Grn]
                              , [GM].[Invoice]
                        FROM    [GrnMatching] [GM]
            End
```

Author: Johnson, Chris

```
    End';
    Declare @SQLInvJournalDet Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS'' and Is-
Numeric(Replace(Db_Name(),''SysproCompany'',''''))=1
    BEGIN
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
        + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables
        + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'',''))
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
                Insert  [#InvJournalDet]
                        ( [DatabaseName]
                        , [JnlYear]
                        , [GlPeriod]
                        , [Journal]
                        , [EntryNumber]
                        , [Supplier]
                        , [PurchaseOrder]
                        , [Reference]
                        )
                        SELECT  [DatabaseName] = @DBCode
                              , [IJD].[JnlYear]
                              , [IJD].[GlPeriod]
                              , [IJD].[Journal]
                              , [IJD].[EntryNumber]
                              , [IJD].[Supplier]
                              , [IJD].[PurchaseOrder]
                              , [IJD].[Reference]
                        FROM    [InvJournalDet] [IJD]
            End
    End';
    Declare @SQLAnalysis Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS'' and Is-
Numeric(Replace(Db_Name(),''SysproCompany'',''''))=1
    BEGIN
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
        + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables
        + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')
```

Author: Johnson, Chris

```
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'',''))
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
                    Insert  [#GenAnalysisTrn]
                            ( [DatabaseName]
                            , [AnalysisEntry]
                            , [GlPeriod]
                            , [GlYear]
                            , [AnalysisCategory]
                            , [AnalysisCode1]
                            , [AnalysisCode2]
                            , [AnalysisCode3]
                            , [AnalysisCode4]
                            , [AnalysisCode5]
                            )
                            Select  [DatabaseName] = @DBCode
                                  , [GAT].[AnalysisEntry]
                                  , [GAT].[GlPeriod]
                                  , [GAT].[GlYear]
                                  , [GAT].[AnalysisCategory]
                                  , [AnalysisCode1] = Case When [GAT].[Analysis-
Code1] = ''''
                                                           Then Null
                                                           Else [GAT].[Analysis-
Code1]
                                                      End
                                  , [AnalysisCode2] = Case When [GAT].[Analysis-
Code2] = ''''
                                                           Then Null
                                                           Else [GAT].[Analysis-
Code2]
                                                      End
                                  , [AnalysisCode3] = Case When [GAT].[Analysis-
Code3] = ''''
                                                           Then Null
                                                           Else [GAT].[Analysis-
Code3]
                                                      End
                                  , [AnalysisCode4] = Case When [GAT].[Analysis-
Code4] = ''''
                                                           Then Null
                                                           Else [GAT].[Analysis-
Code4]
                                                      End
                                  , [AnalysisCode5] = Case When [GAT].[Analysis-
Code5] = ''''
                                                           Then Null
                                                           Else [GAT].[Analysis-
Code5]
                                                      End
                            From    [dbo].[GenAnalysisTrn] [GAT];

                    Insert  [#GenAnalysisCode]
                            ( [DatabaseName]
                            , [AnalysisCategory]
                            , [AnalysisCode]
                            , [AnalysisType]
                            , [Description]
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_Gen-JournalDetails

```
                            )
                            Select  [DatabaseName] = @DBCode
                                  , [GAC].[AnalysisCategory]
                                  , [GAC].[AnalysisCode]
                                  , [GAC].[AnalysisType]
                                  , [GAC].[Description]
                            From    [SysproCompany10].[dbo].[GenAnalysisCode] [GAC];
        End
End';

Print @SQLGenJournalDetail

Exec [Process].[ExecForEachDB] @cmd = @SQLGenJournalDetail;
Exec [Process].[ExecForEachDB] @cmd = @SQLGrnMatching;
Exec [Process].[ExecForEachDB] @cmd = @SQLInvJournalDet;
Exec [Process].[ExecForEachDB] @cmd = @SQLAnalysis;

Select  [Co] = [GJD].[DatabaseName]
      , [CN].[CompanyName]
      , [CN].[ShortName]
      , [GJD].[Journal]
      , [GJD].[GlYear]
      , [GJD].[GlPeriod]
      , [GJD].[EntryNumber]
      , [GJD].[EntryType]
      , [GJD].[GlCode]
      , [GLDescription] = [GM2].[Description]
      , [GJD].[Reference]
      , [GJD].[Comment]
      , [GJD].[EntryValue]
      , [GJD].[EntryDate]
      , [GJD].[EntryPosted]
      , [GJD].[SourceDetail]
      , [GJD].[InterCompanyFlag]
      , [GJD].[Company]
      , [GJD].[CurrencyValue]
      , [GJD].[PostCurrency]
      , [GJD].[TypeDetail]
      , [GJD].[CommitmentFlag]
      , [GJD].[TransactionDate]
      , [GJD].[DocumentDate]
      , [Supplier] = Case When [IJD].[Supplier] = '' Then Null
                          Else [IJD].[Supplier]
                     End
      , [PurchaseOrder] = Case When [IJD].[PurchaseOrder] = '' Then Null
                               Else [IJD].[PurchaseOrder]
                          End
      , [GM].[Grn]
      , [GM].[Invoice]
      , [GAT].[AnalysisCategory]
      , [GAT].[AnalysisCode1]
      , [Analysis1] = [GAC].[Description]
      , [GAT].[AnalysisCode2]
      , [GAT].[AnalysisCode3]
      , [GAT].[AnalysisCode4]
      , [GAT].[AnalysisCode5]
From    [#GenJournalDetail] As [GJD]
```

Author: Johnson, Chris

```
            Left Join [#InvJournalDet] [IJD]
                On [IJD].[JnlYear] = [GJD].[GlYear]
                   And [IJD].[GlPeriod] = [GJD].[GlPeriod]
                   And [IJD].[Journal] = [GJD].[SubModJournal]
                   And [IJD].[EntryNumber] = [GJD].[EntryNumber]
                   And [IJD].[DatabaseName] = [GJD].[DatabaseName]
            Left Join [#GrnMatching] [GM]
                On [IJD].[Reference] = [GM].[Grn]
                   And [GM].[DatabaseName] = [IJD].[DatabaseName]
            Left Join [Lookups].[CompanyNames] As [CN]
                On [CN].[Company] = [GJD].[DatabaseName]
            Left Join [SysproCompany40].[dbo].[GenMaster] [GM2]
                On [GM2].[Company] = [GJD].[DatabaseName]
                   And [GM2].[GlCode] = [GJD].[GlCode]
            Left Join [#GenAnalysisTrn] [GAT]
                On [GAT].[AnalysisEntry] = [GJD].[AnalysisEntry]
                   And [GAT].[GlPeriod] = [GJD].[GlPeriod]
                   And [GAT].[GlYear] = [GJD].[GlYear]
                   And [GAT].[DatabaseName] = [GJD].[DatabaseName]
            Left Join [#GenAnalysisCode] [GAC]
                On [GAC].[AnalysisCategory] = [GAT].[AnalysisCategory]
                   And [GAC].[AnalysisCode] = [GAT].[AnalysisCode1]
                   And [GAC].[AnalysisType] = 1
                   And [GAC].[DatabaseName] = [GAT].[DatabaseName];

    Drop Table [#GenJournalDetail];
    Drop Table [#GrnMatching];
    Drop Table [#InvJournalDet];
    Drop Table [#GenAnalysisCode];
    Drop Table  [#GenAnalysisTrn];
GO
EXEC sp_addextendedproperty N'MS_Description', N'list of journals', 'SCHEMA',
N'Report', 'PROCEDURE', N'UspResults_GenJournalDetails', NULL, NULL
GO
```

Uses

[Lookups].[CompanyNames]
[Process].[ExecForEachDB]
[Process].[UspInsert_RedTagLogs]
[Report]

Author: Johnson, Chris

[Report].[UspResults_GenJournalEntries]

MS_Description

list of journal entries

Parameters

Name	Data Type	Max Length (Bytes)
@Company	varchar(max)	max
@RedTagType	char	1
@RedTagUse	varchar(500)	500

SQL Script

```
CREATE Proc [Report].[UspResults_GenJournalEntries]
    (
      @Company Varchar(Max)
    , @RedTagType Char(1)
    , @RedTagUse Varchar(500)
    )
As
    Begin
/*
Template designed by Chris Johnson, Prometic Group May 2016
Stored procedure set out to query multiple databases with the same information and
return it in a collated format
*/
Set NoCount On

        If IsNumeric(@Company) = 0
            Begin
                Select  @Company = Upper(@Company);
            End;

--Red tag

        Declare @RedTagDB Varchar(255)= Db_Name();
        Exec [Process].[UspInsert_RedTagLogs] @StoredProcDb = 'BlackBox' ,
            @StoredProcSchema = 'Report' ,
            @StoredProcName = 'UspResults_GenJournalEntries' ,
            @UsedByType = @RedTagType , @UsedByName = @RedTagUse ,
            @UsedByDb = @RedTagDB;

--list the tables that are to be pulled back from each DB - if they are not found
the script will not be run against that db

        Declare @ListOfTables Varchar(Max) = 'AssetDepreciation,TblApTerms';

--create temporary tables to be pulled from different databases, including a column
to id
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_Gen-JournalEntries

```
        Create Table [#GenJournalDetail]
            (
              [DatabaseName] Varchar(150)        collate Latin1_General_BIN
            , [Journal] Int
            , [GlYear] Int
            , [GlPeriod] Int
            , [EntryNumber] Int
            , [EntryType] Char(1)                collate Latin1_General_BIN
            , [GlCode] Varchar(35)               collate Latin1_General_BIN
            , [Reference] Varchar(50)            collate Latin1_General_BIN
            , [Comment] Varchar(250)             collate Latin1_General_BIN
            , [EntryValue] Numeric(20 , 2)
            , [EntryDate] Date
            , [EntryPosted] Char(1)              collate Latin1_General_BIN
            , [InterCompanyFlag] Char(1)         collate Latin1_General_BIN
            , [Company] Varchar(50)              collate Latin1_General_BIN
            , [CurrencyValue] Numeric(20 , 2)
            , [PostCurrency] Varchar(10)         collate Latin1_General_BIN
            , [TypeDetail] Varchar(100)          collate Latin1_General_BIN
            , [CommitmentFlag] Char(1)           collate Latin1_General_BIN
            , [TransactionDate] Date
            , [DocumentDate] Date
            , [SubModJournal] Int
            , [AnalysisEntry] Int
            );
        Create Table [#GenJournalCtl]
            (
              [DatabaseName] Varchar(150)        collate Latin1_General_BIN
            , [JnlPrintFlag] Char(1)             collate Latin1_General_BIN
            , [JournalDate] Date
            , [NumOfEntries] Int
            , [DebitAmount] Numeric(20 , 2)
            , [CreditAmount] Numeric(20 , 2)
            , [JnlPostingType] Char(1)           collate Latin1_General_BIN
            , [Source] Char(1)                   collate Latin1_General_BIN
            , [Operator] Varchar(20)             collate Latin1_General_BIN
            , [JnlStatus] Char(1)                collate Latin1_General_BIN
            , [Reference] Varchar(30)            collate Latin1_General_BIN
            , [AuthorisedBy] Varchar(20)         collate Latin1_General_BIN
            , [PostedBy] Varchar(20)             collate Latin1_General_BIN
            , [Authorised] Char(1)               collate Latin1_General_BIN
            , [PostDate] Date
            , [Notation] Varchar(100)            collate Latin1_General_BIN
            , [GlJournal] Int
            , [GlPeriod] Int
            , [GlYear] Int
            , [JournalSource] Char(2)            collate Latin1_General_BIN
            );
        Create Table [#GrnMatching]
            (
              [DatabaseName] Varchar(300)        collate Latin1_General_BIN
            , [Grn] Varchar(20)                  collate Latin1_General_BIN
            , [Invoice] Varchar(20)              collate Latin1_General_BIN
            );
        Create Table [#InvJournalDet]
            (
              [DatabaseName] Varchar(300)        collate Latin1_General_BIN
```

Author: Johnson, Chris

```
            , [JnlYear] Int
            , [GlPeriod] Int
            , [Journal] Int
            , [EntryNumber] Int
            , [Supplier] Varchar(15)            collate Latin1_General_BIN
            , [PurchaseOrder] Varchar(20)        collate Latin1_General_BIN
            , [Reference] Varchar(30)            collate Latin1_General_BIN
            );
        Create Table [#GenAnalysisTrn]
            (
              [DatabaseName] Varchar(150)        collate Latin1_General_BIN
            , [AnalysisEntry] Int
            , [GlPeriod] Int
            , [GlYear] Int
            , [AnalysisCategory] Varchar(10)    collate Latin1_General_BIN
            , [AnalysisCode1] Varchar(10)        collate Latin1_General_BIN
            , [AnalysisCode2] Varchar(10)        collate Latin1_General_BIN
            , [AnalysisCode3] Varchar(10)        collate Latin1_General_BIN
            , [AnalysisCode4] Varchar(10)        collate Latin1_General_BIN
            , [AnalysisCode5] Varchar(10)        collate Latin1_General_BIN
            , [EntryValue] Numeric(20 , 2)
            );
        Create Table [#GenAnalysisCode]
            (
              [DatabaseName] Varchar(150)        collate Latin1_General_BIN
            , [AnalysisCategory] Varchar(10)    collate Latin1_General_BIN
            , [AnalysisCode] Varchar(10)        collate Latin1_General_BIN
            , [AnalysisType] Int
            , [Description] Varchar(50)            collate Latin1_General_BIN
            );

--create script to pull data from each db into the tables

        Declare @SQLGenJournalDetail Varchar(Max) = 'USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS'' and Is-
Numeric(Replace(Db_Name(),''SysproCompany'',''''))=1
    BEGIN
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables
            + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'',''))
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
            Declare @SubSQL Varchar(1000)

            Select @SubSQL=''SELECT [DatabaseName]=''+@DBCode+''
                           , [GJD].[Journal]
                           , [GJD].[GlYear]
```

```
                              , [GJD].[GlPeriod]
                              , [GJD].[EntryNumber]
                              , [GJD].[EntryType]
                              , [GJD].[GlCode]
                              , [GJD].[Reference]
                              , [GJD].[Comment]
                              , [GJD].[EntryValue]
                              , [GJD].[EntryDate]
                              , [GJD].[EntryPosted]
                              , [GJD].[InterCompanyFlag]
                              , [GJD].[Company]
                              , [GJD].[CurrencyValue]
                              , [GJD].[PostCurrency]
                              , [TypeDetail] = Coalesce([GJT].[TypeDetail],''''No
Type'''')
                              , [GJD].[CommitmentFlag]
                              , [GJD].[TransactionDate]
                              , [GJD].[DocumentDate]
                              , [GJD].[SubModJournal]
                              , [GJD].[AnalysisEntry] FROM [GenJournalDetail] As [GJD]
            Left Join [BlackBox].[Lookups].[GenJournalDetailSource] As [GJDS] On
[GJDS].[GJSource]=[GJD].[Source]
            Left Join [BlackBox].[Lookups].[GenJournalType] As [GJT] On
[GJD].[Type]=[GJT].[TypeCode]''

                    Insert [#GenJournalDetail]
                              ( [DatabaseName]
                              , [Journal]
                              , [GlYear]
                              , [GlPeriod]
                              , [EntryNumber]
                              , [EntryType]
                              , [GlCode]
                              , [Reference]
                              , [Comment]
                              , [EntryValue]
                              , [EntryDate]
                              , [EntryPosted]
                              , [InterCompanyFlag]
                              , [Company]
                              , [CurrencyValue]
                              , [PostCurrency]
                              , [TypeDetail]
                              , [CommitmentFlag]
                              , [TransactionDate]
                              , [DocumentDate]
                              , [SubModJournal]
                              , [AnalysisEntry]
                              )
                    Exec (@SubSQL)
            End
    End';
        Declare @SQLGenJournalCtl Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS'' and Is-
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_Gen-JournalEntries

```
Numeric(Replace(Db_Name(),''SysproCompany'','''))=1
    BEGIN
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables
            + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'',''))
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
                    Insert [#GenJournalCtl]
                            ( [DatabaseName]
                            , [JnlPrintFlag]
                            , [JournalDate]
                            , [NumOfEntries]
                            , [DebitAmount]
                            , [CreditAmount]
                            , [JnlPostingType]
                            , [Source]
                            , [Operator]
                            , [JnlStatus]
                            , [Reference]
                            , [AuthorisedBy]
                            , [PostedBy]
                            , [Authorised]
                            , [PostDate]
                            , [Notation]
                            , [GlJournal]
                            , [GlPeriod]
                            , [GlYear]
                            , [JournalSource]
                            )
                    SELECT [DatabaseName]=@DBCode
                         , [GJC].[JnlPrintFlag]
                         , [GJC].[JournalDate]
                         , [GJC].[NumOfEntries]
                         , [GJC].[DebitAmount]
                         , [GJC].[CreditAmount]
                         , [GJC].[JnlPostingType]
                         , [GJC].[Source]
                         , [GJC].[Operator]
                         , [GJC].[JnlStatus]
                         , [GJC].[Reference]
                         , [GJC].[AuthorisedBy]
                         , [GJC].[PostedBy]
                         , [GJC].[Authorised]
                         , [GJC].[PostDate]
                         , [GJC].[Notation]
                         , [GJC].[GlJournal]
                         , [GJC].[GlPeriod]
                         , [GJC].[GlYear]
                         , [GJC].[JournalSource] FROM [GenJournalCtl] As [GJC]
```

Author: Johnson, Chris

```
            End
    End';
        Declare @SQLGrnMatching Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS'' and Is-
Numeric(Replace(Db_Name(),''SysproCompany'',''''))=1
    BEGIN
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables
            + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'',''))
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
                Insert [#GrnMatching]
                ( [DatabaseName] , [Grn] , [Invoice] )
                SELECT [DatabaseName]=@DBCode
                , [GM].[Grn]
                , [GM].[Invoice]
                FROM [GrnMatching] [GM]
            End
    End';
        Declare @SQLInvJournalDet Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS'' and Is-
Numeric(Replace(Db_Name(),''SysproCompany'',''''))=1
    BEGIN
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables
            + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'',''))
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
                Insert [#InvJournalDet]
                        ( [DatabaseName]
                        , [JnlYear]
                        , [GlPeriod]
                        , [Journal]
                        , [EntryNumber]
                        , [Supplier]
```

Author: Johnson, Chris

```
                        , [PurchaseOrder]
                        , [Reference]
                        )
                SELECT [DatabaseName]=@DBCode
                     , [IJD].[JnlYear]
                     , [IJD].[GlPeriod]
                     , [IJD].[Journal]
                     , [IJD].[EntryNumber]
                     , [IJD].[Supplier]
                     , [IJD].[PurchaseOrder]
                     , [IJD].[Reference]
                FROM [InvJournalDet] [IJD]
            End
    End';
        Declare @SQLAnalysis Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS'' and Is-
Numeric(Replace(Db_Name(),''SysproCompany'',''''))=1
    BEGIN
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables
            + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'',''))
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
                    Insert  [#GenAnalysisTrn]
                            ( [DatabaseName]
                            , [AnalysisEntry]
                            , [GlPeriod]
                            , [GlYear]
                            , [AnalysisCategory]
                            , [AnalysisCode1]
                            , [AnalysisCode2]
                            , [AnalysisCode3]
                            , [AnalysisCode4]
                            , [AnalysisCode5]
                            , [EntryValue]
                            )
                            Select  [DatabaseName] = @DBCode
                                  , [GAT].[AnalysisEntry]
                                  , [GAT].[GlPeriod]
                                  , [GAT].[GlYear]
                                  , [GAT].[AnalysisCategory]
                                  , [AnalysisCode1] = Case When [GAT].[Analysis-
Code1] = ''''
                                                           Then Null
                                                           Else [GAT].[Analysis-
Code1]
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_Gen-JournalEntries

```
                                                End
                                    , [AnalysisCode2] = Case When [GAT].[Analysis-
Code2] = ''''
                                                         Then Null
                                                         Else [GAT].[Analysis-
Code2]
                                                End
                                    , [AnalysisCode3] = Case When [GAT].[Analysis-
Code3] = ''''
                                                         Then Null
                                                         Else [GAT].[Analysis-
Code3]
                                                End
                                    , [AnalysisCode4] = Case When [GAT].[Analysis-
Code4] = ''''
                                                         Then Null
                                                         Else [GAT].[Analysis-
Code4]
                                                End
                                    , [AnalysisCode5] = Case When [GAT].[Analysis-
Code5] = ''''
                                                         Then Null
                                                         Else [GAT].[Analysis-
Code5]
                                                End
                                    , [GAT].[EntryValue]
                            From    [dbo].[GenAnalysisTrn] [GAT];

                    Insert  [#GenAnalysisCode]
                            ( [DatabaseName]
                            , [AnalysisCategory]
                            , [AnalysisCode]
                            , [AnalysisType]
                            , [Description]
                            )
                            Select  [DatabaseName] = @DBCode
                                  , [GAC].[AnalysisCategory]
                                  , [GAC].[AnalysisCode]
                                  , [GAC].[AnalysisType]
                                  , [GAC].[Description]
                            From    [dbo].[GenAnalysisCode] [GAC];
            End
    End';

--execute script against each db, populating the base tables

        Exec [Process].[ExecForEachDB] @cmd = @SQLGenJournalCtl;
        Exec [Process].[ExecForEachDB] @cmd = @SQLGenJournalDetail;
        Exec [Process].[ExecForEachDB] @cmd = @SQLGrnMatching;
        Exec [Process].[ExecForEachDB] @cmd = @SQLInvJournalDet;
        Exec [Process].[ExecForEachDB] @cmd = @SQLAnalysis;

--define the results you want to return

        Create Table [#Results]
            (
              [DatabaseName] Varchar(150)        collate Latin1_General_BIN
            , [Journal] Int
            , [GlYear] Int
```

Author: Johnson, Chris

```
            , [GlPeriod] Int
            , [EntryNumber] Int
            , [EntryType] Char(1)                collate Latin1_General_BIN
            , [GlCode] Varchar(35)               collate Latin1_General_BIN
            , [Reference] Varchar(50)            collate Latin1_General_BIN
            , [Comment] Varchar(250)             collate Latin1_General_BIN
            , [EntryValue] Numeric(20 , 2)
            , [EntryDate] Date
            , [EntryPosted] Char(1)                collate Latin1_General_BIN
            , [JnlPrintFlag] Char(1)             collate Latin1_General_BIN
            , [JournalDate] Date
            , [NumOfEntries] Int
            , [DebitAmount] Numeric(20 , 2)
            , [CreditAmount] Numeric(20 , 2)
            , [JnlPostingType] Char(1)             collate Latin1_General_BIN
            , [Source] Char(1)                     collate Latin1_General_BIN
            , [Operator] Varchar(20)             collate Latin1_General_BIN
            , [JnlStatus] Char(1)                collate Latin1_General_BIN
            , [AuthorisedBy] Varchar(20)         collate Latin1_General_BIN
            , [PostedBy] Varchar(20)             collate Latin1_General_BIN
            , [Authorised] Char(1)                 collate Latin1_General_BIN
            , [PostDate] Date
            , [Notation] Varchar(100)            collate Latin1_General_BIN
            , [JournalSource] Char(2)            collate Latin1_General_BIN
            , [Supplier] Varchar(15)             collate Latin1_General_BIN
            , [PurchaseOrder] Varchar(20)        collate Latin1_General_BIN
            , [Grn] Varchar(Max)                 collate Latin1_General_BIN
            , [Invoice] Varchar(Max)             collate Latin1_General_BIN
            , [AnalysisCategory] Varchar(10)     collate Latin1_General_BIN
            , [AnalysisCode1] Varchar(10)        collate Latin1_General_BIN
            , [Analysis1] Varchar(50)            collate Latin1_General_BIN
            , [AnalysisCode2] Varchar(10)        collate Latin1_General_BIN
            , [AnalysisCode3] Varchar(10)        collate Latin1_General_BIN
            , [AnalysisCode4] Varchar(10)        collate Latin1_General_BIN
            , [AnalysisCode5] Varchar(10)        collate Latin1_General_BIN
            , [AnalysisEntryValue] Numeric(20 , 2)
            );

--Placeholder to create indexes as required


--script to combine base data and insert into results table

        Insert  [#Results]
                ( [DatabaseName]
                , [Journal]
                , [GlYear]
                , [GlPeriod]
                , [EntryNumber]
                , [EntryType]
                , [GlCode]
                , [Reference]
                , [Comment]
                , [EntryValue]
                , [EntryDate]
                , [EntryPosted]
                , [JnlPrintFlag]
```

Author: Johnson, Chris

```
                , [JournalDate]
                , [NumOfEntries]
                , [DebitAmount]
                , [CreditAmount]
                , [JnlPostingType]
                , [Source]
                , [Operator]
                , [JnlStatus]
                , [AuthorisedBy]
                , [PostedBy]
                , [Authorised]
                , [PostDate]
                , [Notation]
                , [JournalSource]
                , [Supplier]
                , [PurchaseOrder]
                , [Grn]
                , [Invoice]
                , [AnalysisCategory]
                , [AnalysisCode1]
                , [Analysis1]
                , [AnalysisCode2]
                , [AnalysisCode3]
                , [AnalysisCode4]
                , [AnalysisCode5]
                , [AnalysisEntryValue]
                )
                Select  [GJD].[DatabaseName]
                      , [GJD].[Journal]
                      , [GJD].[GlYear]
                      , [GJD].[GlPeriod]
                      , [GJD].[EntryNumber]
                      , [GJD].[EntryType]
                      , [GJD].[GlCode]
                      , [GJD].[Reference]
                      , [GJD].[Comment]
                      , [GJD].[EntryValue]
                      , [GJD].[EntryDate]
                      , [GJD].[EntryPosted]
                      , [GJC].[JnlPrintFlag]
                      , [GJC].[JournalDate]
                      , [GJC].[NumOfEntries]
                      , [GJC].[DebitAmount]
                      , [GJC].[CreditAmount]
                      , [GJC].[JnlPostingType]
                      , [GJC].[Source]
                      , [GJC].[Operator]
                      , [GJC].[JnlStatus]
                      , [GJC].[AuthorisedBy]
                      , [GJC].[PostedBy]
                      , [GJC].[Authorised]
                      , [GJC].[PostDate]
                      , [GJC].[Notation]
                      , [GJC].[JournalSource]
                      , [Supplier] = Case When [IJD].[Supplier] = '' Then Null
                                          Else [IJD].[Supplier]
                                     End
```

Author: Johnson, Chris

```
                    , [PurchaseOrder] = Case When [IJD].[PurchaseOrder] = ''
                                             Then Null
                                             Else [IJD].[PurchaseOrder]
                                        End
                    , [Grn] = Stuff(( Select Distinct
                                              ',' + [GM].[Grn]
                                      From    [#GrnMatching] [GM]
                                      Where   [IJD].[Reference] = [GM].[Grn]
                                              And [GM].[DatabaseName] =
[IJD].[DatabaseName]
                                              And Coalesce([GM].[Invoice] ,
                                                           '') <> ''
                                    For
                                      Xml Path('')
                                    ) , 1 , 1 , '')
                    , [Invoice] = Stuff(( Select Distinct
                                                 ',' + [GM].[Invoice]
                                          From    [#GrnMatching] [GM]
                                          Where   [IJD].[Reference] = [GM].[Grn]
                                                  And [GM].[DatabaseName] =
[IJD].[DatabaseName]
                                                  And Coalesce([GM].[Invoice] ,
                                                               '') <> ''
                                        For
                                          Xml Path('')
                                        ) , 1 , 1 , '')
                    , [GAT].[AnalysisCategory]
                    , [GAT].[AnalysisCode1]
                    , [Analysis1] = [GAC].[Description]
                    , [GAT].[AnalysisCode2]
                    , [GAT].[AnalysisCode3]
                    , [GAT].[AnalysisCode4]
                    , [GAT].[AnalysisCode5]
                    , [GAT].[EntryValue]
                From    [#GenJournalDetail] As [GJD]
                  Left Join [#GenJournalCtl] As [GJC]
                      On [GJC].[GlJournal] = [GJD].[Journal]
                         And [GJC].[GlPeriod] = [GJD].[GlPeriod]
                         And [GJC].[GlYear] = [GJD].[GlYear]
                         And [GJC].[DatabaseName] = [GJD].[DatabaseName]
                  Left Join [#InvJournalDet] [IJD]
                      On [IJD].[JnlYear] = [GJD].[GlYear]
                         And [IJD].[GlPeriod] = [GJD].[GlPeriod]
                         And [IJD].[Journal] = [GJD].[SubModJournal]
                         And [IJD].[EntryNumber] = [GJD].[EntryNumber]
                         And [IJD].[DatabaseName] = [GJD].[DatabaseName]
                  /*Left Join [#GrnMatching] [GM]
                      On [IJD].[Reference] = [GM].[Grn]
                         And [GM].[DatabaseName] = [IJD].[DatabaseName]*/
                  Left Join [Lookups].[CompanyNames] As [CN]
                      On [CN].[Company] = [GJD].[DatabaseName]
                  Left Join [#GenAnalysisTrn] [GAT]
                      On [GAT].[AnalysisEntry] = [GJD].[AnalysisEntry]
                         And [GAT].[GlPeriod] = [GJD].[GlPeriod]
                         And [GAT].[GlYear] = [GJD].[GlYear]
                         And [GAT].[DatabaseName] = [GJD].[DatabaseName]
                  Left Join [#GenAnalysisCode] [GAC]
```

```
                            On [GAC].[AnalysisCategory] = [GAT].[AnalysisCategory]
                               And [GAC].[AnalysisCode] = [GAT].[AnalysisCode1]
                               And [GAC].[AnalysisType] = 1
                               And [GAC].[DatabaseName] = [GAT].[DatabaseName];

Set NoCount Off

--return results

        Select  [R].[DatabaseName]
              , [CN].[CompanyName]
              , [R].[Journal]
              , [R].[GlYear]
              , [R].[GlPeriod]
              , [R].[EntryNumber]
              , [R].[EntryType]
              , [R].[GlCode]
              , [GLDescription] = [GM].[Description]
              , [R].[Reference]
              , [R].[Comment]
              , [R].[EntryValue]
              , [R].[EntryDate]
              , [R].[EntryPosted]
              , [R].[JnlPrintFlag]
              , [R].[JournalDate]
              , [R].[NumOfEntries]
              , [R].[DebitAmount]
              , [R].[CreditAmount]
              , [JPT].[JnlPostingTypeDesc]
              , [GJCS].[SourceDescription]
              , [R].[Operator]
              , [JS].[JnlStatusDesc]
              , [R].[AuthorisedBy]
              , [R].[PostedBy]
              , [R].[Authorised]
              , [R].[PostDate]
              , [R].[Notation]
              , [JournalSource] = [GJCJS].[GenJournalCtlJnlSourceDesc]
              , [R].[Supplier]
              , [R].[PurchaseOrder]
              , [R].[Grn]
              , [R].[Invoice]
              , [R].[AnalysisCategory]
              , [R].[AnalysisCode1]
              , [R].[Analysis1]
              , [R].[AnalysisCode2]
              , [R].[AnalysisCode3]
              , [R].[AnalysisCode4]
              , [R].[AnalysisCode5]
              , [R].[AnalysisEntryValue]
        From    [#Results] As [R]
                Left Join [BlackBox].[Lookups].[JnlPostingType] [JPT]
                    On [R].[JnlPostingType] = [JPT].[JnlPostingType]
                Left Join [BlackBox].[Lookups].[JnlStatus] [JS]
                    On [R].[JnlStatus] = [JS].[JnlStatus]
                Left Join [BlackBox].[Lookups].[GenJournalCtlSource] [GJCS]
                    On [R].[Source] = [GJCS].[Source]
```

Author: Johnson, Chris

```
                Left Join [SysproCompany40].[dbo].[GenMaster] As [GM]
                    On [GM].[GlCode] = [R].[GlCode]
                       And [GM].[Company] = [R].[DatabaseName]
                Left Join [BlackBox].[Lookups].[CompanyNames] As [CN]
                    On [CN].[Company] = [R].[DatabaseName]
                Left Join [Lookups].[GenJournalCtlJnlSource] [GJCJS]
                    On [R].[JournalSource] = [GJCJS].[GenJournalCtlJnlSource];

    End;

GO
EXEC sp_addextendedproperty N'MS_Description', N'list of journal entries', 'SCHEMA',
N'Report', 'PROCEDURE', N'UspResults_GenJournalEntries', NULL, NULL
GO
```

Uses

[Lookups].[CompanyNames]
[Lookups].[GenJournalCtlJnlSource]
[Lookups].[GenJournalCtlSource]
[Lookups].[JnlPostingType]
[Lookups].[JnlStatus]
[Process].[ExecForEachDB]
[Process].[UspInsert_RedTagLogs]
[Report]

[Report].[UspResults_GenledgerJournalsGrouped]

MS_Description

list of journals grouped

Parameters

Name	Data Type	Max Length (Bytes)
@Company	varchar(max)	max
@RedTagType	char	1
@RedTagUse	varchar(500)	500

SQL Script

```
CREATE Proc [Report].[UspResults_GenledgerJournalsGrouped]
    (
      @Company Varchar(Max)
    , @RedTagType Char(1)
    , @RedTagUse Varchar(500)
    )
As
    Begin
/*
Template designed by Chris Johnson, Prometic Group September 2015
Stored procedure set out to query multiple databases with the same information and
return it in a collated format
--Exec [Report].[UspResults_GenledgerJournalsGrouped] @Company ='10'
*/
        If IsNumeric(@Company) = 0
            Begin
                Select  @Company = Upper(@Company);
            End;

--remove nocount on to speed up query

        Set NoCount Off;
--Red tag

        Declare @RedTagDB Varchar(255)= Db_Name();
        Exec [Process].[UspInsert_RedTagLogs] @StoredProcDb = 'BlackBox' ,
            @StoredProcSchema = 'Report' ,
            @StoredProcName = 'UspResults_GenledgerJournalsGrouped' ,
            @UsedByType = @RedTagType , @UsedByName = @RedTagUse ,
            @UsedByDb = @RedTagDB;

--list the tables that are to be pulled back from each DB - if they are not found
the script will not be run against that db

        Declare @ListOfTables Varchar(Max) = 'GenJournalDetail';

--create temporary tables to be pulled from different databases, including a column
```

Author: Johnson, Chris

```
to id

        Create Table [#GenJournalDetail]
            (
              [DatabaseName] Varchar(150)        collate Latin1_General_BIN
            , [GlYear] Int
            , [GlPeriod] Int
            , [EntryDate] DateTime2
            , [Journal] Int
            , [GlCode] Varchar(150)             collate Latin1_General_BIN
            , [EntryType] Char(1)                collate Latin1_General_BIN
            , [Reference] Varchar(255)            collate Latin1_General_BIN
            , [Comment] Varchar(255)            collate Latin1_General_BIN
            , [EntryValue] Numeric(20 , 7)
            , [EntryPosted] Char(1)                collate Latin1_General_BIN
            , [SubModJournal] Int
            , [Description] Varchar(150)        collate Latin1_General_BIN
            );

--create script to pull data from each db into the tables

        Declare @SQL Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end'
            + --Only query DBs beginning SysProCompany

            '
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN'
            + --only companies selected in main run, or if companies selected then
all

            '
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT'
            + --count number of tables requested (number of commas plus one)

            '
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')'
            + --Count of the tables requested how many exist in the db

            '
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'','')) '
            + --only if the count matches (all the tables exist in the requested db)
then run the script

            '
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
                Insert [#GenJournalDetail]
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-GenledgerJournalsGrouped

```
                        ( [DatabaseName], [GlYear]
                        , [GlPeriod], [EntryDate]
                        , [Journal], [GlCode]
                        , [EntryType], [Reference]
                        , [Comment], [EntryValue]
                        , [EntryPosted], [SubModJournal]
                        ,[Description]
                        )
                SELECT [DatabaseName]=@DBCode, [gjd].[GlYear]
                     , [gjd].[GlPeriod], [gjd].[EntryDate]
                     , [gjd].[Journal], [gjd].[GlCode]
                     , [gjd].[EntryType], [gjd].[Reference]
                     , [gjd].[Comment], [gjd].[EntryValue]
                     , [gjd].[EntryPosted], [gjd].[SubModJournal]
                     , gm.[Description]
                From [GenJournalDetail] As [gjd]
                left join [SysproCompany40]..[GenMaster] As [gm]
                            on gjd.GlCode=gm.GlCode
            End
    End';
--Enable this function to check script changes (try to run script directly against
db manually)

--Print @SQL


--execute script against each db, populating the base tables

        Exec [Process].[ExecForEachDB] @cmd = @SQL;

--define the results you want to return

        Create Table [#ResultsGL]
            (
              [DatabaseName] Varchar(150)        collate Latin1_General_BIN
            , [Mapping1] Varchar(255)            collate Latin1_General_BIN
            , [Mapping2] Varchar(255)            collate Latin1_General_BIN
            , [Mapping3] Varchar(255)            collate Latin1_General_BIN
            , [Mapping4] Varchar(255)            collate Latin1_General_BIN
            , [Mapping5] Varchar(255)            collate Latin1_General_BIN
            , [GlYear] Int
            , [GlPeriod] Int
            , [EntryDate] DateTime2
            , [Journal] Int
            , [GlCode] Varchar(150)                collate Latin1_General_BIN
            , [GlStart] Char(3)                    collate Latin1_General_BIN
            , [GlMid] Char(5)                    collate Latin1_General_BIN
            , [GlEnd] Char(3)                    collate Latin1_General_BIN
            , [EntryType] Char(1)                collate Latin1_General_BIN
            , [Modifier] Numeric(20 , 7)
            , [Reference] Varchar(250)
            , [Comment] Varchar(250)
            , [EntryValue] Numeric(20 , 7)
            , [EntryPosted] Char(1)                collate Latin1_General_BIN
            , [SubModJournal] Int
            , [GlDescription] Varchar(150)        collate Latin1_General_BIN
            );
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-GenledgerJournalsGrouped

```
--Placeholder to create indexes as required


--script to combine base data and insert into results table

        Insert  [#ResultsGL]
                ( [DatabaseName]
                , [Mapping1]
                , [Mapping2]
                , [Mapping3]
                , [Mapping4]
                , [Mapping5]
                , [GlYear]
                , [GlPeriod]
                , [EntryDate]
                , [Journal]
                , [GlCode]
                , [GlStart]
                , [GlMid]
                , [GlEnd]
                , [EntryType]
                , [Modifier]
                , [Reference]
                , [Comment]
                , [EntryValue]
                , [EntryPosted]
                , [SubModJournal]
                , [GlDescription]
                )
                Select  Coalesce([gd].[DatabaseName] , [gm].[Company])
                      , [Mapping1] = Coalesce([gm].[Mapping1] , 'No map')
                      , [Mapping2] = Coalesce([gm].[Mapping2] , 'No map')
                      , [Mapping3] = Coalesce([gm].[Mapping3] , 'No map')
                      , [Mapping4] = Coalesce([gm].[Mapping4] , 'No map')
                      , [Mapping5] = Coalesce([gm].[Mapping5] , 'No map')
                      , [gd].[GlYear]
                      , [gd].[GlPeriod]
                      , [gd].[EntryDate]
                      , [gd].[Journal]
                      , [GlCode] = Coalesce([gd].[GlCode] , [gm].[GlCode])
                      , [GlStart] = Cast(Coalesce([gm].[GlStart] ,
                                                  ParseName([gd].[GlCode] , 3)) As
Char(3))
                      , [GlMid] = Cast(Coalesce([gm].[GlMid] ,
                                                ParseName([gd].[GlCode] , 2)) As
Char(5))
                      , [GlEnd] = Cast(Coalesce([gm].[GlEnd] ,
                                                ParseName([gd].[GlCode] , 1)) As
Char(3))
                      , [gd].[EntryType]
                      , [Modifier] = Case When [gd].[EntryType] = 'D' Then 1
                                          When [gd].[EntryType] = 'C' Then -1
                                          Else 0
                                     End
                      , [Reference] = Case When [gd].[Reference] = ''
                                           Then Null
                                           Else [gd].[Reference]
```

Author: Johnson, Chris

```
                                    End
                      , [Comment] = Case When [gd].[Comment] = '' Then Null
                                         Else [gd].[Comment]
                                    End
                      , [EntryValue] = Coalesce([gd].[EntryValue] , 0)
                      , [gd].[EntryPosted]
                      , [SubModJournal] = Case When [gd].[SubModJournal] = 0
                                               Then Null
                                               Else [gd].[SubModJournal]
                                          End
                      , [gd].[Description]
                from    [Lookups].[GLMapping] As [gm]
                        Full Outer Join [#GenJournalDetail] [gd] On [gd].[GlCode] =
[gm].[GlCode]
                                                                    And [gd].[Database-
Name] = [gm].[Company];

--return results

        Select  [Company] = [rg].[DatabaseName]
              , [cn].[CompanyName]
              , [rg].[Mapping1]
              , [rg].[Mapping2]
              , [rg].[Mapping3]
              , [rg].[Mapping4]
              , [rg].[Mapping5]
              , [rg].[GlYear]
              , [rg].[GlPeriod]
              , [EntryDate] = Cast([rg].[EntryDate] As Date)
              , [rg].[Journal]
              , [rg].[GlCode]
              , [rg].[GlStart]
              , [rg].[GlMid]
              , [rg].[GlEnd]
              , [rg].[EntryType]
              , [rg].[Modifier]
              , [rg].[Reference]
              , [rg].[Comment]
              , [rg].[EntryValue]
              , [rg].[EntryPosted]
              , [rg].[SubModJournal]
              , [rg].[GlDescription]
        From    [#ResultsGL] As [rg]
                Left Join [Lookups].[CompanyNames] [cn] On [rg].[DatabaseName] =
[cn].[Company];

    End;

GO
EXEC sp_addextendedproperty N'MS_Description', N'list of journals grouped',
'SCHEMA', N'Report', 'PROCEDURE', N'UspResults_GenledgerJournalsGrouped', NULL, NULL
GO
```

Uses

[Lookups].[CompanyNames]

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-GenledgerJournalsGrouped

[Lookups].[GLMapping]

[Process].[ExecForEachDB]

[Process].[UspInsert_RedTagLogs]

[Report]

Author: Johnson, Chris

[Report].[UspResults_GenMasterMapping]

MS_Description

not in use

Parameters

Name	Data Type	Max Length (Bytes)
@Company	varchar(max)	max
@RedTagType	char	1
@RedTagUse	varchar(500)	500

SQL Script

```
CREATE Proc [Report].[UspResults_GenMasterMapping]
    (
      @Company Varchar(Max)
    , @RedTagType Char(1)
    , @RedTagUse Varchar(500)
    )
--Exec [Report].[UspResults_GenMasterMapping] @Company='10'

As
    Begin
/*
Template designed by Chris Johnson, Prometic Group September 2015
Stored procedure set out to query multiple databases with the same information and
return it in a collated format
*/
        If IsNumeric(@Company) = 0
            Begin
                Select  @Company = Upper(@Company);
            End;

--remove nocount on to speed up query

        Set NoCount On;

--Red tag

        Declare @RedTagDB Varchar(255)= Db_Name();
        Exec [Process].[UspInsert_RedTagLogs] @StoredProcDb = 'BlackBox' ,
            @StoredProcSchema = 'Report' ,
            @StoredProcName = 'UspResults_GenMasterMapping' ,
            @UsedByType = @RedTagType , @UsedByName = @RedTagUse ,
            @UsedByDb = @RedTagDB;

--create temporary tables to be pulled from different databases, including a column
to id

        Create Table [#GenMaster]
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_Gen-MasterMapping

```
            (
              [Company] Varchar(150)        Collate Latin1_General_BIN
            , [GlCode] Varchar(150)            Collate Latin1_General_BIN
            , [Description] Varchar(255)    Collate Latin1_General_BIN
            );
        Print 1;
        Insert  [#GenMaster]
                ( [Company]
                , [GlCode]
                , [Description]
                )
                Select  [gm].[Company]
                      , [gm].[GlCode]
                      , [gm].[Description]
                From    [SysproCompany40]..[GenMaster] As [gm]
                Where   [gm].[Company] = @Company
                        And [gm].[GlCode] Not In ( 'FORCED' , 'RETAINED' );



--define the results you want to return

        Create Table [#Results]
            (
              [GlCode] Varchar(150)                    collate Latin1_General_BIN
            , [Company] Varchar(150)                collate Latin1_General_BIN
            , [GlStart] Varchar(3)                    collate Latin1_General_BIN
            , [GlMid] Varchar(5)                    collate Latin1_General_BIN
            , [Mid1] Varchar(3)                        collate Latin1_General_BIN
            , [Mid2] Varchar(2)                        collate Latin1_General_BIN
            , [GlEnd] Varchar(3)                    collate Latin1_General_BIN
            , [MappingDescription] Varchar(255)        collate Latin1_General_BIN
            , [LedgerDescription] Varchar(255)        collate Latin1_General_BIN
            , [Mapping1] Varchar(255)                collate Latin1_General_BIN
            , [Mapping2] Varchar(255)                collate Latin1_General_BIN
            , [Mapping3] Varchar(255)                collate Latin1_General_BIN
            , [Mapping4] Varchar(255)                collate Latin1_General_BIN
            , [Mapping5] Varchar(255)                collate Latin1_General_BIN
            , [Status] Varchar(255)                    collate Latin1_General_BIN
            );

        Print 2;
--script to combine base data and insert into results table

        Insert  [#Results]
                ( [GlCode]
                , [Company]
                , [GlStart]
                , [GlMid]
                , [Mid1]
                , [Mid2]
                , [GlEnd]
                , [MappingDescription]
                , [LedgerDescription]
                , [Mapping1]
                , [Mapping2]
                , [Mapping3]
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_Gen-MasterMapping

```
                , [Mapping4]
                , [Mapping5]
                , [Status]
                )
                Select  [GlCode] = Coalesce([gm].[GlCode] , [gm2].[GlCode])
                      , [Company] = Coalesce([gm].[Company] , [gm2].[Company])
                      , [GlStart] = Coalesce([gm2].[GlStart] ,
                                             ParseName([gm].[GlCode] , 3))
                      , [GlMid] = Coalesce([gm2].[GlMid] ,
                                           ParseName([gm].[GlCode] , 2))
                      , [Mid1] = Coalesce([gm2].[Mid1] ,
                                          Left(( ParseName([gm].[GlCode] , 2) ) ,
                                               3))
                      , [Mid2] = Coalesce([gm2].[Mid2] ,
                                          Right(( ParseName([gm].[GlCode] , 2) ) ,
                                                2))
                      , [GlEnd] = Coalesce([gm2].[GlEnd] ,
                                           ParseName([gm].[GlCode] , 1))
                      , [MappingDescription] = [gm2].[GlDescription]
                      , [LedgerDescription] = [gm].[Description]
                      , [Mapping1] = [gm2].[Mapping1]
                      , [Mapping2] = [gm2].[Mapping2]
                      , [Mapping3] = [gm2].[Mapping3]
                      , [Mapping4] = [gm2].[Mapping4]
                      , [Mapping5] = [gm2].[Mapping5]
                      , [Status] = Case When [gm].[GlCode] Is     Null
                                        Then 'Map does not have value in General
Ledger' Collate Latin1_General_BIN
                                        When [gm2].[GlCode] Is     Null
                                        Then 'Ledger code does not have a map'
Collate Latin1_General_BIN
                                        When [gm2].[GlDescription] <>
[gm].[Description]
                                        Then 'Map description does not match GL
description' Collate Latin1_General_BIN
                                        When [gm].[GlCode] Is Not Null
                                             And [gm2].[GlCode] Is Not Null
                                        Then 'Map Available' Collate Latin1_General_-
BIN
                                   End
                  From    [#GenMaster] As [gm]
                          Full Outer Join [Lookups].[GLMapping] As [gm2] On
[gm2].[Company] = [gm].[Company]
                                                                  And [gm2].[GlCode] =
[gm].[GlCode];

--return results

        Select  [GlCode]
              , [Company]
              , [GlStart]
              , [GlMid]
              , [Mid1]
              , [Mid2]
              , [GlEnd]
              , [MappingDescription]
              , [LedgerDescription]
              , [Mapping1]
              , [Mapping2]
```

Author: Johnson, Chris

```
                , [Mapping3]
                , [Mapping4]
                , [Mapping5]
                , [Status]
        From    [#Results];

    End;

GO
EXEC sp_addextendedproperty N'MS_Description', N'not in use', 'SCHEMA', N'Report',
'PROCEDURE', N'UspResults_GenMasterMapping', NULL, NULL
GO
```

Uses

[Lookups].[GLMapping]
[Process].[UspInsert_RedTagLogs]
[Report]

Author: Johnson, Chris

[Report].[UspResults_GL_Actuals]

MS_Description

list of gl actual figures

SQL Script

```
CREATE Proc [Report].[UspResults_GL_Actuals]
As
/*
////////////////////////////////////////////////////////////////////////////////////////////////////
///            Template designed by Chris Johnson, Prometic Group September 2015
///
///
///
///            This returns the difference between the start and end of each month
as it is captured in the Gen Ledger history table    ///
///
///
///            Version 1.0.1
///
///
///
///            Change Log
///
///
///
///            Date        Person                    Description
///
///            7/9/2015    Chris Johnson            Initial version created
///
///            7/9/2015    Chris Johnson            Changed to use of udf_Split-
String to define tables to return                      ///
///            4/1/2016    Chris Johnson            Added Movement to date for use
in trial balance                                   ///
///            ??/??/201?    Placeholder                Placeholder
///
///            ??/??/201?    Placeholder                Placeholder
///
///            ??/??/201?    Placeholder                Placeholder
///
///            ??/??/201?    Placeholder                Placeholder
///
////////////////////////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////
*/


    Begin
        Select  Company
              , GlCode
              , Period = Cast(Replace(Period , 'Period' , '') As Int)
              , Actual
              , YearMovement
              , GlYear
              , ClosingBalance
              , MovementToDate
```

Author: Johnson, Chris

```
        From    ( Select    GH.Company
                          , GH.GlCode
                          , GH.GlYear
                          , Period1                = GH.ClosingBalPer1  - GH.Begin-
YearBalance
                          , Period2                = GH.ClosingBalPer2  - GH.Closing-
BalPer1
                          , Period3                = GH.ClosingBalPer3  - GH.Closing-
BalPer2
                          , Period4                = GH.ClosingBalPer4  - GH.Closing-
BalPer3
                          , Period5                = GH.ClosingBalPer5  - GH.Closing-
BalPer4
                          , Period6                = GH.ClosingBalPer6  - GH.Closing-
BalPer5
                          , Period7                = GH.ClosingBalPer7  - GH.Closing-
BalPer6
                          , Period8                = GH.ClosingBalPer8  - GH.Closing-
BalPer7
                          , Period9                = GH.ClosingBalPer9  - GH.Closing-
BalPer8
                          , Period10            = GH.ClosingBalPer10 - GH.ClosingBal-
Per9
                          , Period11            = GH.ClosingBalPer11 - GH.ClosingBal-
Per10
                          , Period12            = GH.ClosingBalPer12 - GH.ClosingBal-
Per11
                          , GH.BeginYearBalance
                          , [YearMovement]        = ClosingBalPer13    - BeginYear-
Balance
                          , GH.ClosingBalPer1
                          , GH.ClosingBalPer2
                          , GH.ClosingBalPer3
                          , GH.ClosingBalPer4
                          , GH.ClosingBalPer5
                          , GH.ClosingBalPer6
                          , GH.ClosingBalPer7
                          , GH.ClosingBalPer8
                          , GH.ClosingBalPer9
                          , GH.ClosingBalPer10
                          , GH.ClosingBalPer11
                          , GH.ClosingBalPer12
                          , MovementToDate1               = GH.ClosingBalPer1  -
GH.BeginYearBalance
                          , MovementToDate2               = GH.ClosingBalPer2  -
GH.BeginYearBalance
                          , MovementToDate3               = GH.ClosingBalPer3  -
GH.BeginYearBalance
                          , MovementToDate4               = GH.ClosingBalPer4  -
GH.BeginYearBalance
                          , MovementToDate5               = GH.ClosingBalPer5  -
GH.BeginYearBalance
                          , MovementToDate6               = GH.ClosingBalPer6  -
GH.BeginYearBalance
                          , MovementToDate7               = GH.ClosingBalPer7  -
GH.BeginYearBalance
                          , MovementToDate8               = GH.ClosingBalPer8  -
GH.BeginYearBalance
                          , MovementToDate9               = GH.ClosingBalPer9  -
GH.BeginYearBalance
                          , MovementToDate10           = GH.ClosingBalPer10 -
GH.BeginYearBalance
                          , MovementToDate11           = GH.ClosingBalPer11 -
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_GL_-Actuals

```
GH.BeginYearBalance
                          , MovementToDate12            = GH.ClosingBalPer12 -
GH.BeginYearBalance
                  From      SysproCompany40.dbo.GenHistory GH
                ) Actuals Unpivot ( Actual For Period In ( Period1 , Period2 ,
                                                           Period3 , Period4 ,
                                                           Period5 , Period6 ,
                                                           Period7 , Period8 ,
                                                           Period9 , Period10,
                                                           Period11, Period12 ) ) As
ASMT
                        Unpivot (ClosingBalance For Period2 In (ClosingBalPer1 ,
ClosingBalPer2
                                                                , ClosingBalPer3 ,
ClosingBalPer4
                                                                , ClosingBalPer5 ,
ClosingBalPer6
                                                                , ClosingBalPer7 ,
ClosingBalPer8
                                                                , ClosingBalPer9 ,
ClosingBalPer10
                                                                , ClosingBalPer11,
ClosingBalPer12) )As CBP
                        Unpivot (MovementToDate For Period3 In (MovementToDate1,
MovementToDate2
                                                                , MovementToDate3,
MovementToDate4
                                                                , MovementToDate5,
MovementToDate6
                                                                , MovementToDate7,
MovementToDate8
                                                                , MovementToDate9,
MovementToDate10
                                                                , MovementToDate11,
MovementToDate12) )As MTD
                Where Cast(Replace(Period , 'Period' , '') As
Int)=Cast(Replace(Period2 , 'ClosingBalPer' , '') As Int)
                And Cast(Replace(Period , 'Period' , '') As
Int)=Cast(Replace(Period3 , 'MovementToDate' , '') As Int);

--SELECT * FROM #Results As R


    End;
GO
EXEC sp_addextendedproperty N'MS_Description', N'list of gl actual figures',
'SCHEMA', N'Report', 'PROCEDURE', N'UspResults_GL_Actuals', NULL, NULL
GO
```

Uses

[Report]

Used By

[Report].[UspResults_BudgetsActuals]

Author: Johnson, Chris

[Report].[UspResults_GL_Budgets]

MS_Description

list of gl budget figures

SQL Script

```
Create Proc [Report].[UspResults_GL_Budgets]
As
    Begin
                Select  Gb.Company
                      , Gb.GlCode
                      , Budget
                      , BudgetType
                      , Period = Replace(Period,'Budget','')
                FROM SysproCompany40.dbo.GenBudgets Unpivot (Budget For Period In
(Budget1
                      , Budget2
                      , Budget3
                      , Budget4
                      , Budget5
                      , Budget6
                      , Budget7
                      , Budget8
                      , Budget9
                      , Budget10
                      , Budget11
                      , Budget12)) As Gb
                Where BudgetType='C'

    End
GO
EXEC sp_addextendedproperty N'MS_Description', N'list of gl budget figures',
'SCHEMA', N'Report', 'PROCEDURE', N'UspResults_GL_Budgets', NULL, NULL
GO
```

Uses

[Report]

Used By

[Report].[UspResults_BudgetsActuals]

Author: Johnson, Chris

[Report].[UspResults_GL_Codes]

MS_Description

list of gl codes

Parameters

Name	Data Type	Max Length (Bytes)
@RedTagType	char	1
@RedTagUse	varchar(500)	500

SQL Script

```
CREATE Proc [Report].[UspResults_GL_Codes]
    (
      @RedTagType Char(1)
    , @RedTagUse Varchar(500)
    )
As
    Begin

--remove nocount on to speed up query

        Set NoCount On;

--Red tag

        Declare @RedTagDB Varchar(255)= Db_Name();
        Exec [Process].[UspInsert_RedTagLogs] @StoredProcDb = 'BlackBox' ,
            @StoredProcSchema = 'Report' ,
            @StoredProcName = 'UspResults_GL_CodeLists' ,
            @UsedByType = @RedTagType , @UsedByName = @RedTagUse ,
            @UsedByDb = @RedTagDB;


--define the results you want to return

        Create Table [#Results]
            (
              [Company] Char(4)                    collate latin1_general_bin
            , [GlCode] Varchar(35)                 collate latin1_general_bin
            , [Description] Varchar(50)            collate latin1_general_bin
            , [ReportIndex1] Varchar(35)           collate latin1_general_bin
            , [ReportIndex2] Varchar(35)           collate latin1_general_bin
            , [GlGroup] Varchar(10)                collate latin1_general_bin
            , [CurrentBalance] Numeric(20 , 2)
            , [PrevPerEndBal] Numeric(20 , 2)
            );

--script to combine base data and insert into results table
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_GL_-Codes

```
        Insert  [#Results]
                ( [Company]
                , [GlCode]
                , [Description]
                , [ReportIndex1]
                , [ReportIndex2]
                , [GlGroup]
                , [CurrentBalance]
                , [PrevPerEndBal]
                )
                Select  [GM].[Company]
                      , [GM].[GlCode]
                      , [GM].[Description]
                      , [GM].[ReportIndex1]
                      , [GM].[ReportIndex2]
                      , [GM].[GlGroup]
                      , [GM].[CurrentBalance]
                      , [GM].[PrevPerEndBal]
                From    [SysproCompany40].[dbo].[GenMaster] [GM]
                Where   Upper([GM].[GlCode]) Not In ( 'FORCED' , 'RETAINED' );

Set NoCount Off
--return results

        Select  [R].[Company]
              , [R].[GlCode]
              , [R].[Description]
              , [R].[ReportIndex1]
              , [R].[ReportIndex2]
              , [R].[GlGroup]
              , [R].[CurrentBalance]
              , [R].[PrevPerEndBal]
              , [CN].[CompanyName]
              , [CN].[ShortName]
              , [CN].[Currency]
        From    [#Results] [R]
        Left Join [Lookups].[CompanyNames] [CN] On [CN].[Company] = [R].[Company];

    End;

GO
EXEC sp_addextendedproperty N'MS_Description', N'list of gl codes', 'SCHEMA',
N'Report', 'PROCEDURE', N'UspResults_GL_Codes', NULL, NULL
GO
```

Uses

[Lookups].[CompanyNames]
[Process].[UspInsert_RedTagLogs]
[Report]

Author: Johnson, Chris

[Report].[UspResults_GLMovements]

MS_Description

list of gl movements

Parameters

Name	Data Type	Max Length (Bytes)
@RedTagType	char	1
@RedTagUse	varchar(500)	500

SQL Script

```
CREATE Proc [Report].[UspResults_GLMovements]
    (
      @RedTagType Char(1)
    , @RedTagUse Varchar(500)
    )
As
    Begin

        Set NoCount On;
--Red tag

        Declare @RedTagDB Varchar(255)= Db_Name();
        Exec [Process].[UspInsert_RedTagLogs] @StoredProcDb = 'BlackBox' ,
            @StoredProcSchema = 'Report' ,
            @StoredProcName = 'UspResults_GLMovements' ,
            @UsedByType = @RedTagType , @UsedByName = @RedTagUse ,
            @UsedByDb = @RedTagDB;

        Declare @GlYearPeriod Int;

        Create Table [#YearPeriod]
            (
              [GlYear] Int
            , [GlPeriod] Int
            );

        Insert  [#YearPeriod]
                ( [GlYear]
                , [GlPeriod]
                )
                Select Distinct
                        [GT].[GlYear]
                      , [P].[Number]
                From    [SysproCompany40]..[GenTransaction] [GT]
                        Cross Join ( Select [Number]
                                     From   [dbo].[UdfResults_NumberRange](0 ,
                                                                           15)
                                   ) [P];
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-GLMovements

```
        Create Table [#Movements]
            (
              [Company] Varchar(10) Collate Latin1_General_BIN
            , [ShortName] Varchar(250) Collate Latin1_General_BIN
            , [CompanyName] Varchar(250) Collate Latin1_General_BIN
            , [Currency] Varchar(10) Collate Latin1_General_BIN
            , [GlCode] Varchar(35) Collate Latin1_General_BIN
            , [Description] Varchar(50) Collate Latin1_General_BIN
            , [GlGroup] Varchar(10) Collate Latin1_General_BIN
            , [Movement] Numeric(20 , 2)
            , [GlPeriod] Int
            , [GlYear] Int
            , [Source] Varchar(100) Collate Latin1_General_BIN
            , [Journal] Int
            , [ReportIndex1] Varchar(35) Collate Latin1_General_BIN
            , [ReportIndex2] Varchar(35) Collate Latin1_General_BIN
            , [GLAccountTypeDesc] Varchar(250) Collate Latin1_General_BIN
            );
        Create Table [#MovementsRaw]
            (
              [Company] Varchar(10) Collate Latin1_General_BIN
            , [ShortName] Varchar(250) Collate Latin1_General_BIN
            , [CompanyName] Varchar(250) Collate Latin1_General_BIN
            , [Currency] Varchar(10) Collate Latin1_General_BIN
            , [GlCode] Varchar(35) Collate Latin1_General_BIN
            , [Description] Varchar(50) Collate Latin1_General_BIN
            , [GlGroup] Varchar(10) Collate Latin1_General_BIN
            , [Movement] Numeric(20 , 2)
            , [GlPeriod] Int
            , [GlYear] Int
            , [ReportIndex1] Varchar(35) Collate Latin1_General_BIN
            , [ReportIndex2] Varchar(35) Collate Latin1_General_BIN
            , [GLAccountTypeDesc] Varchar(250) Collate Latin1_General_BIN
            );

        Insert  [#Movements]
                ( [Company]
                , [ShortName]
                , [CompanyName]
                , [Currency]
                , [GlCode]
                , [Description]
                , [GlGroup]
                , [Movement]
                , [GlPeriod]
                , [GlYear]
                , [Source]
                , [Journal]
                , [ReportIndex1]
                , [ReportIndex2]
                , [GLAccountTypeDesc]
                )
                Select  [t].[Company]
                      , [t].[ShortName]
                      , [t].[CompanyName]
                      , [t].[Currency]
```

Author: Johnson, Chris

```
                        , [t].[GlCode]
                        , [t].[Description]
                        , [t].[GlGroup]
                        , [t].[Movement]
                        , [t].[GlPeriod]
                        , [t].[GlYear]
                        , [t].[Source]
                        , [t].[Journal]
                        , [t].[ReportIndex1]
                        , [t].[ReportIndex2]
                        , [t].[GLAccountTypeDesc]
                From    ( Select    [GM].[Company]
                                  , [CN].[ShortName]
                                  , [CN].[CompanyName]
                                  , [CN].[Currency]
                                  , [GM].[GlCode]
                                  , [GM].[Description]
                                  , [GM].[GlGroup]
                                  , [Movement] = ( [GH].[BeginYearBalance] )
                                  , [GlPeriod] = 0
                                  , [GH].[GlYear]
                                  , [Source] = 'History'
                                  , [Journal] = 0
                                  , [GM].[ReportIndex1]
                                  , [GM].[ReportIndex2]
                                  , [GAT].[GLAccountTypeDesc]
                          From      [SysproCompany40].dbo.[GenMaster] As [GM]
                                    Left Join [BlackBox].[Lookups].[CompanyNames]
                                        As [CN]
                                        On [CN].[Company] = [GM].[Company]
                                    Left Join [SysproCompany40].[dbo].[GenHistory]
[GH]
                                        On [GH].[Company] = [GM].[Company]
                                           And [GH].[GlCode] = [GM].[GlCode]
                                    Left Join [BlackBox].[Lookups].[GLAccountType]
[GAT]
                                        On [GAT].[GLAccountType] = [GM].[Account-
Type]
                          Where     [GH].[GlYear] >= 2013
                          Union All
                          Select    [GM].[Company]
                                  , [CN].[ShortName]
                                  , [CN].[CompanyName]
                                  , [CN].[Currency]
                                  , [GM].[GlCode]
                                  , [GM].[Description]
                                  , [GM].[GlGroup]
                                  , [Movement] = [GT].[EntryValue]
                                  , [GT].[GlPeriod]
                                  , [GT].[GlYear]
                                  , [Source] = 'Transactions'
                                  , [GT].[Journal]
                                  , [GM].[ReportIndex1]
                                  , [GM].[ReportIndex2]
                                  , [GAT].[GLAccountTypeDesc]
                          From      [SysproCompany40]..[GenMaster] As [GM]
                                    Left Join [SysproCompany40].[dbo].[Gen-
Transaction] [GT]
```

Author: Johnson, Chris

```
                                        On [GT].[Company] = [GM].[Company]
                                           And [GT].[GlCode] = [GM].[GlCode]
                                    Left Join [BlackBox].[Lookups].[CompanyNames]
                                        As [CN]
                                        On [CN].[Company] = [GM].[Company]
                                    Left Join [BlackBox].[Lookups].[GLAccountType]
[GAT]
                                        On [GAT].[GLAccountType] = [GM].[Account-
Type]
                             Where      [GT].[EntryValue] <> 0
                           ) [t]
                  Order By [t].[Company]
                         , [t].[GlCode]
                         , [t].[GlYear]
                         , [t].[GlPeriod];

        --Get latest Gl period
        Select  @GlYearPeriod = Max(( [M].[GlYear] * 100 ) + [M].[GlPeriod])
        From    [#Movements] [M];

        Insert  [#MovementsRaw]
                ( [Company]
                , [ShortName]
                , [CompanyName]
                , [Currency]
                , [GlCode]
                , [Description]
                , [GlGroup]
                , [Movement]
                , [GlPeriod]
                , [GlYear]
                , [ReportIndex1]
                , [ReportIndex2]
                , [GLAccountTypeDesc]
                )
                Select  [GM].[Company]
                      , [CN].[ShortName]
                      , [CN].[CompanyName]
                      , [CN].[Currency]
                      , [GM].[GlCode]
                      , [GM].[Description]
                      , [GM].[GlGroup]
                      , [Movement] = Convert(Numeric(20 , 4) , 0)
                      , [YP].[GlPeriod]
                      , [YP].[GlYear]
                      , [GM].[ReportIndex1]
                      , [GM].[ReportIndex2]
                      , [GAT].[GLAccountTypeDesc]
                From    [SysproCompany40].dbo.[GenMaster] [GM]
                        Left Join [BlackBox].[Lookups].[CompanyNames] [CN]
                            On [CN].[Company] = [GM].[Company]
                        Cross Join [#YearPeriod] [YP]
                        Left Join [BlackBox].[Lookups].[GLAccountType] [GAT]
                            On [GAT].[GLAccountType] = [GM].[AccountType];

        Insert  [#Movements]
                ( [Company]
```

```
                , [ShortName]
                , [CompanyName]
                , [Currency]
                , [GlCode]
                , [Description]
                , [GlGroup]
                , [Movement]
                , [GlPeriod]
                , [GlYear]
                , [Source]
                , [Journal]
                , [ReportIndex1]
                , [ReportIndex2]
                , [GLAccountTypeDesc]
                )
                Select  [MR].[Company]
                      , [MR].[ShortName]
                      , [MR].[CompanyName]
                      , [MR].[Currency]
                      , [MR].[GlCode]
                      , [MR].[Description]
                      , [MR].[GlGroup]
                      , [MR].[Movement]
                      , [MR].[GlPeriod]
                      , [MR].[GlYear]
                      , [Source] = 'Generated'
                      , [Journal] = 0
                      , [M].[ReportIndex1]
                      , [M].[ReportIndex2]
                      , [M].[GLAccountTypeDesc]
                From    [#MovementsRaw] [MR]
                        Left Join [#Movements] [M]
                            On [M].[Company] = [MR].[Company]
                               And [M].[GlCode] = [MR].[GlCode]
                               And [M].[GlYear] = [MR].[GlYear]
                               And [M].[GlPeriod] = [MR].[GlPeriod]
                Where   [M].[Company] Is Null;

        --Remove null periods and generate

        Delete  [#Movements]
        Where   ( [GlYear] * 100 ) + [GlPeriod] > @GlYearPeriod
                Or [GlPeriod] Is Null;

        Set NoCount Off
        Select  [M].[Company]
              , [M].[ShortName]
              , [M].[CompanyName]
              , [M].[Currency]
              , [M].[GlCode]
              , [M].[Description]
              , [M].[GlGroup]
              , [M].[Movement]
              , [M].[GlPeriod]
              , [M].[GlYear]
              , [M].[Source]
              , [M].[Journal]
```

Author: Johnson, Chris

```
                , [M].[ReportIndex1]
                , [M].[ReportIndex2]
                , AccountType = [M].[GLAccountTypeDesc]
                , [Parse1] = ParseName([M].[GlCode],1)
                , [Parse2] = ParseName([M].[GlCode],2)
                , [Parse3] = ParseName([M].[GlCode],3)
        From    [#Movements] [M]
        Order By [M].[ShortName]
                , [M].[GlCode]
                , [M].[GlYear] Asc
                , [M].[GlPeriod] Asc;

        Drop Table [#YearPeriod];
        Drop Table [#Movements];
        Drop Table [#MovementsRaw];

    End;

GO
EXEC sp_addextendedproperty N'MS_Description', N'list of gl movements', 'SCHEMA',
N'Report', 'PROCEDURE', N'UspResults_GLMovements', NULL, NULL
GO
```

Uses

[Lookups].[CompanyNames]
[Lookups].[GLAccountType]
[Process].[UspInsert_RedTagLogs]
[dbo].[UdfResults_NumberRange]
[Report]

Used By

[Report].[UspResults_GLMovementsIntercoPL]
[Report].[UspResults_GLMovementsPBLRevCos]
[Report].[UspResults_MovementsTrialBalances]

Author: Johnson, Chris

[Report].[UspResults_GLMovementsIntercoPL]

MS_Description

Intercompany profit and loss general ledger

Parameters

Name	Data Type	Max Length (Bytes)
@RedTagType	char	1
@RedTagUse	varchar(500)	500

SQL Script

```
CREATE Proc [Report].[UspResults_GLMovementsIntercoPL]
    (
      @RedTagType Char(1)
    , @RedTagUse Varchar(500)
    )
As
    Begin

        Set NoCount On;
        --Red tag

        Declare @RedTagDB Varchar(255)= Db_Name();
        Exec [Process].[UspInsert_RedTagLogs] @StoredProcDb = 'BlackBox' ,
            @StoredProcSchema = 'Report' ,
            @StoredProcName = 'UspResults_GLMovementsIntercoPL' ,
            @UsedByType = @RedTagType , @UsedByName = @RedTagUse ,
            @UsedByDb = @RedTagDB;

        Create Table [#Movements]
            (
              [Company] Varchar(10) Collate Latin1_General_BIN
            , [ShortName] Varchar(250) Collate Latin1_General_BIN
            , [CompanyName] Varchar(250) Collate Latin1_General_BIN
            , [Currency] Varchar(10) Collate Latin1_General_BIN
            , [GlCode] Varchar(35) Collate Latin1_General_BIN
            , [Description] Varchar(50) Collate Latin1_General_BIN
            , [GlGroup] Varchar(10) Collate Latin1_General_BIN
            , [Movement] Numeric(20 , 2)
            , [GlPeriod] Int
            , [GlYear] Int
            , [Source] Varchar(100) Collate Latin1_General_BIN
            , [Journal] Int
            , [ReportIndex1] Varchar(100) Collate Latin1_General_BIN
            , [ReportIndex2] Varchar(100) Collate Latin1_General_BIN
            , [AccountType] Varchar(100) Collate Latin1_General_BIN
            , [Parse1] Varchar(100) Collate Latin1_General_BIN
            , [Parse2] Varchar(100) Collate Latin1_General_BIN
            , [Parse3] Varchar(100) Collate Latin1_General_BIN
```

Author: Johnson, Chris

```
            );

        Insert  [#Movements]
                ( [Company]
                , [ShortName]
                , [CompanyName]
                , [Currency]
                , [GlCode]
                , [Description]
                , [GlGroup]
                , [Movement]
                , [GlPeriod]
                , [GlYear]
                , [Source]
                , [Journal]
                , [ReportIndex1]
                , [ReportIndex2]
                , [AccountType]
                , [Parse1]
                , [Parse2]
                , [Parse3]
                )
                Exec [Report].[UspResults_GLMovements] @RedTagType = @RedTagType , --
- char(1)

                    @RedTagUse = @RedTagUse;

        Select  [M].[Company]
              , [M].[ShortName]
              , [M].[CompanyName]
              , [M].[Currency]
              , [M].[GlCode]
              , [M].[Description]
              , [M].[GlGroup]
              , [M].[Movement]
              , [M].[GlPeriod]
              , [M].[GlYear]
              , [M].[Source]
              , [M].[Journal]
              , [DueToCo] = Right([M].[GlCode] , 2)
              , [DueToShortName] = Coalesce([CN].[ShortName] , 'Unknown')
              , [DueToCompanyName] = Coalesce([CN].[CompanyName] , 'Unknown')
              , [DateForCurrency] = Convert(DateTime , Case When [M].[GlPeriod] > 12
                                                            Then DateFrom-
Parts([M].[GlYear] ,
                                                              12 , 31)
                                                            When [M].[GlPeriod] = 0
                                                            Then DateFromParts((
[M].[GlYear] ) ,
                                                              1 , 31)
                                                            Else DateFrom-
Parts([M].[GlYear] ,
                                                              [M].[GlPeriod] ,
                                                              28)
                                                       End)
        From    [#Movements] [M]
                Left Join [Lookups].[CompanyNames] [CN]
                    On [CN].[Company] = Right([M].[GlCode] , 2)
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-GLMovementsIntercoPL

```
        Where   [M].[GlGroup] In ( 'INTERCOREV' , 'INTERCOEXP' , 'RLTYINTRCO' ,
                                   'INTINTERCO' , 'INTERCOCOS' );


    End;
GO
EXEC sp_addextendedproperty N'MS_Description', N'Intercompany profit and loss
general ledger', 'SCHEMA', N'Report', 'PROCEDURE', N'UspResults_GLMovementsInterco-
PL', NULL, NULL
GO
```

Uses

[Lookups].[CompanyNames]
[Process].[UspInsert_RedTagLogs]
[Report].[UspResults_GLMovements]
[Report]

Author: Johnson, Chris

[Report].[UspResults_GLMovementsPBLRevCos]

MS_Description

PBL Revenue and Cost of Sales

Parameters

Name	Data Type	Max Length (Bytes)
@RedTagType	char	1
@RedTagUse	varchar(500)	500

SQL Script

```
CREATE Proc [Report].[UspResults_GLMovementsPBLRevCos]
    (
      @RedTagType Char(1)
    , @RedTagUse Varchar(500)
    )
As
    Begin

        Set NoCount On;
        --Red tag

        Declare @RedTagDB Varchar(255)= Db_Name();
        Exec [Process].[UspInsert_RedTagLogs] @StoredProcDb = 'BlackBox' ,
            @StoredProcSchema = 'Report' ,
            @StoredProcName = 'UspResults_GLMovementsPBLRevCos' ,
            @UsedByType = @RedTagType , @UsedByName = @RedTagUse ,
            @UsedByDb = @RedTagDB;

        Create Table [#Movements]
            (
              [Company] Varchar(10) Collate Latin1_General_BIN
            , [ShortName] Varchar(250) Collate Latin1_General_BIN
            , [CompanyName] Varchar(250) Collate Latin1_General_BIN
            , [Currency] Varchar(10) Collate Latin1_General_BIN
            , [GlCode] Varchar(35) Collate Latin1_General_BIN
            , [Description] Varchar(50) Collate Latin1_General_BIN
            , [GlGroup] Varchar(10) Collate Latin1_General_BIN
            , [Movement] Numeric(20 , 2)
            , [GlPeriod] Int
            , [GlYear] Int
            , [Source] Varchar(100) Collate Latin1_General_BIN
            , [Journal] Int
            , [ReportIndex1] Varchar(100) Collate Latin1_General_BIN
            , [ReportIndex2] Varchar(100) Collate Latin1_General_BIN
            , [AccountType] Varchar(100) Collate Latin1_General_BIN
            , [Parse1] Varchar(100) Collate Latin1_General_BIN
            , [Parse2] Varchar(100) Collate Latin1_General_BIN
            , [Parse3] Varchar(100) Collate Latin1_General_BIN
```

Author: Johnson, Chris

```
            );

        Insert  [#Movements]
                ( [Company]
                , [ShortName]
                , [CompanyName]
                , [Currency]
                , [GlCode]
                , [Description]
                , [GlGroup]
                , [Movement]
                , [GlPeriod]
                , [GlYear]
                , [Source]
                , [Journal]
                , [ReportIndex1]
                , [ReportIndex2]
                , [AccountType]
                , [Parse1]
                , [Parse2]
                , [Parse3]
                )
                Exec [Report].[UspResults_GLMovements] @RedTagType = @RedTagType ,
                    @RedTagUse = @RedTagUse;

Set NoCount Off

        Select  [t].[Company]
              , [t].[ShortName]
              , [t].[CompanyName]
              , [t].[Currency]
              , [t].[GlCode]
              , [t].[Description]
              , [t].[GlGroup]
              , [t].[Movement]
              , [t].[GlPeriod]
              , [t].[GlYear]
              , [t].[Source]
              , [t].[Journal]
              , [t].[RevGLCode]
              , [RevGLDescription] = [GM].[Description]
        From    ( Select    [M].[Company]
                          , [M].[ShortName]
                          , [M].[CompanyName]
                          , [M].[Currency]
                          , [M].[GlCode]
                          , [M].[Description]
                          , [M].[GlGroup]
                          , [M].[Movement]
                          , [M].[GlPeriod]
                          , [M].[GlYear]
                          , [M].[Source]
                          , [M].[Journal]
                          , [RevGLCode] = [M].[GlCode]
                  From      [#Movements] [M]
                  Where     [M].[GlCode] In ( '101.50000.000' ,
                                              '101.50005.000' ,
```

Author: Johnson, Chris

```
                                        '101.50010.000' ,
                                        '101.50015.000' ,
                                        '101.50020.000' ,
                                        '101.50025.000' ,
                                        '101.50030.000' ,
                                        '101.50035.000' ,
                                        '101.50040.000' ,
                                        '101.50050.000' ,
                                        '101.50050.001' ,
                                        '101.50055.000' ,
                                        '101.50095.000' ,
                                        '101.50095.001' )
                    Union All
                    Select    [M].[Company]
                            , [M].[ShortName]
                            , [M].[CompanyName]
                            , [M].[Currency]
                            , [M].[GlCode]
                            , [M].[Description]
                            , [M].[GlGroup]
                            , [M].[Movement]
                            , [M].[GlPeriod]
                            , [M].[GlYear]
                            , [M].[Source]
                            , [M].[Journal]
                            , [RevGLCode] = Case When [M].[GlCode] = '101.60050.100'
                                                 Then '101.50050.001'
                                                 When [M].[GlCode] = '101.60095.002'
                                                 Then '101.50095.001'
                                                 Else '101.5'
                                                      + Right(ParseName([M].[GlCode] ,
                                                                        2) , 4) + '.000'
                                            End
                    From      [#Movements] [M]
                    Where     [M].[GlCode] In ( '101.60000.000' ,
                                                '101.60000.001' ,
                                                '101.60005.000' ,
                                                '101.60005.001' ,
                                                '101.60010.000' ,
                                                '101.60010.000' ,
                                                '101.60015.000' ,
                                                '101.60015.001' ,
                                                '101.60020.000' ,
                                                '101.60020.001' ,
                                                '101.60025.000' ,
                                                '101.60025.001' ,
                                                '101.60030.000' ,
                                                '101.60030.001' ,
                                                '101.60035.000' ,
                                                '101.60035.001' ,
                                                '101.60040.000' ,
                                                '101.60040.001' ,
                                                '101.60050.000' ,
                                                '101.60050.001' ,
                                                '101.60050.100' ,
                                                '101.60055.000' ,
                                                '101.60055.001' ,
```

```
                                            '101.60095.000' ,
                                            '101.60095.001' ,
                                            '101.60095.002' )
                ) [t]
                Left Join [SysproCompany40].[dbo].[GenMaster] [GM]
                    On [t].[RevGLCode] = [GM].[GlCode]
        Order By [t].[RevGLCode]
              , [t].[GlYear]
              , [t].[GlPeriod];

        Drop Table [#Movements];

    End;
GO
EXEC sp_addextendedproperty N'MS_Description', N'PBL Revenue and Cost of Sales',
'SCHEMA', N'Report', 'PROCEDURE', N'UspResults_GLMovementsPBLRevCos', NULL, NULL
GO
```

Uses

[Process].[UspInsert_RedTagLogs]
[Report].[UspResults_GLMovements]
[Report]

Author: Johnson, Chris

[Report].[UspResults_GrnInvoiceDetails]

MS_Description

provides details of goods received in relation to invoices

Parameters

Name	Data Type	Max Length (Bytes)
@Company	varchar(max)	max
@RedTagType	char	1
@RedTagUse	varchar(500)	500

SQL Script

```
CREATE Proc [Report].[UspResults_GrnInvoiceDetails]
    (
      @Company Varchar(Max)
    , @RedTagType Char(1)
    , @RedTagUse Varchar(500)
    )
As
    Begin
/*
Template designed by Chris Johnson, Prometic Group September 2015
Stored procedure set out to query multiple databases with the same information and
return it in a collated format
 --exec [Report].[UspResults_GrnInvoiceDetails] @Company ='10'
*/
        If IsNumeric(@Company) = 0
            Begin
                Select  @Company = Upper(@Company);
            End;

--remove nocount on to speed up query

        Set NoCount On;

--Red tag

        Declare @RedTagDB Varchar(255)= Db_Name();
        Exec [Process].[UspInsert_RedTagLogs] @StoredProcDb = 'BlackBox' ,
            @StoredProcSchema = 'Report' ,
            @StoredProcName = 'UspResults_GrnInvoiceDetails' ,
            @UsedByType = @RedTagType , @UsedByName = @RedTagUse ,
            @UsedByDb = @RedTagDB;

--list the tables that are to be pulled back from each DB - if they are not found
the script will not be run against that db

        Declare @ListOfTables Varchar(Max) = 'GrnMatching,GrnDetails,PorMaster-
Detail,ReqRouting';
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_Grn-InvoiceDetails

```
        Declare @LoadDate DateTime2 = GetDate()
          , @IsComplete Bit;

--Create Results Table

        If Not Exists ( Select  [T].[name]
                        From    [sys].[schemas] As [S]
                                Left Join [sys].[tables] As [T] On [T].[schema_id] =
[S].[schema_id]
                        Where   [S].[name] = 'Report'
                                And [T].[name] = 'GrnInvoiceDetails' )
            Begin
                Create Table [Report].[GrnInvoiceDetails]
                    (
                      [DatabaseName] Varchar(150)         Collate Latin1_General_BIN
                    , [Supplier] Varchar(150)             Collate Latin1_General_BIN
                    , [Grn] Varchar(50)                     Collate Latin1_General_-
BIN
                    , [TransactionType] Varchar(5)         Collate Latin1_General_-
BIN
                    , [Journal] Int
                    , [EntryNumber] Int
                    , [Invoice] Varchar(150)             Collate Latin1_General_BIN
                    , [PurchaseOrder] Varchar(150)         Collate Latin1_General_-
BIN
                    , [PurchaseOrderLine] Int
                    , [Requisition] Varchar(150)         Collate Latin1_General_BIN
                    , [RequisitionLine] Int
                    , [GlCode] Varchar(50)                 Collate Latin1_General_-
BIN
                    , [Description] Varchar(150)         Collate Latin1_General_BIN
                    , [MatchedValue] Decimal(15 , 3)
                    , [MatchedDate] Date
                    , [StockCode] Varchar(50)             Collate Latin1_General_BIN
                    , [QtyReceived] Decimal(20 , 12)
                    , [MatchedYear] Int
                    , [MatchedMonth] Int
                    , [MatchedQty] Decimal(20 , 12)
                    , [Operator] Varchar(150)             Collate Latin1_General_BIN
                    , [Approver] Varchar(150)             Collate Latin1_General_BIN
                    , [OrigReceiptDate] Date
                    , [LoadDate] DateTime2
                    );
            End;
--Create Process table - this will capture if job is incomplete

        If Not Exists ( Select  [T].[name]
                        From    [sys].[schemas] As [S]
                                Left Join [sys].[tables] As [T] On [T].[schema_id] =
[S].[schema_id]
                        Where   [S].[name] = 'Process'
                                And [T].[name] = 'GrnInvoiceDetails' )
            Begin
                Create Table [Process].[GrnInvoiceDetails]
                    (
                      [LoadDate] DateTime2
                    , [IsComplete] Bit Default 0
                    , [Company] Varchar(150) Collate Latin1_General_BIN
```

Author: Johnson, Chris

```
                    );
            End;

        Select  @IsComplete = [IsComplete]
              , @LoadDate = [LoadDate]
        From    [Process].[GrnInvoiceDetails]
        Where   DateDiff(Minute , [LoadDate] , @LoadDate) <= 3
                And [Company] = @Company;

--if there are no started procs then grab data and store locally

        If @IsComplete Is Null
            Begin
                Insert  [Process].[GrnInvoiceDetails]
                        ( [LoadDate]
                        , [IsComplete]
                        , [Company]
                        )
                Values  ( @LoadDate
                        , 0
                        , @Company
                        );

--create temporary tables to be pulled from different databases, including a column to id

                Create Table [#GrnMatching]
                    (
                      [DatabaseName] Varchar(150)         Collate Latin1_General_BIN
                    , [Supplier] Varchar(150)             Collate Latin1_General_BIN
                    , [Grn] Varchar(50)                     Collate Latin1_General_-
BIN
                    , [TransactionType] Varchar(5)         Collate Latin1_General_-
BIN
                    , [Journal] Int
                    , [EntryNumber] Int
                    , [Invoice] Varchar(50)                 Collate Latin1_General_-
BIN
                    , [MatchedValue] Decimal(15 , 2)
                    , [MatchedDate] DateTime2
                    , [MatchedYear] Int
                    , [MatchedMonth] Int
                    , [MatchedQty] Decimal(20 , 12)
                    );
                Create Table [#GrnDetails]
                    (
                      [DatabaseName] Varchar(150)        Collate Latin1_General_BIN
                    , [PurchaseOrder] Varchar(20)        Collate Latin1_General_BIN
                    , [PurchaseOrderLin] Int
                    , [DebitRecGlCode] Varchar(50)        Collate Latin1_General_BIN
                    , [StockCode] Varchar(30)            Collate Latin1_General_BIN
                    , [QtyReceived] Decimal(25 , 8)
                    , [Supplier] Varchar(150)            Collate Latin1_General_BIN
                    , [Grn] Varchar(25)                     Collate Latin1_General_-
BIN
                    , [GrnSource] Varchar(150)           Collate Latin1_General_BIN
                    , [Journal] Int
                    , [JournalEntry] Int
```

```
                    , [OrigReceiptDate] Date
                    );
                Create Table [#PorMasterDetail]
                    (
                      [DatabaseName] Varchar(150)         Collate Latin1_General_BIN
                    , [MRequisition] Varchar(50)         Collate Latin1_General_BIN
                    , [MRequisitionLine] Int
                    , [PurchaseOrder] Varchar(50)         Collate Latin1_General_BIN
                    , [Line] Int
                    );
                Create Table [#ReqRouting]
                    (
                      [DatabaseName] Varchar(150)         Collate Latin1_General_BIN
                    , [Requisition] Varchar(10)             Collate Latin1_General_-
BIN
                    , [RequisitionLine] Int
                    , [Operator] Varchar(150)             Collate Latin1_General_BIN
                    );
                Create Table [#ReqDetails]
                    (
                      [Requisition] Varchar(10)             Collate Latin1_General_-
BIN
                    , [UserCode] Varchar(20)             Collate Latin1_General_BIN
                    , [DatabaseName] Varchar(150)         Collate Latin1_General_BIN
                    , [Line] Int
                    );

--create script to pull data from each db into the tables

                Declare @SQL1a Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end'
                    + --Only query DBs beginning SysProCompany

                    '
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN'
                    + --only companies selected in main run, or if companies
selected then all

                    '
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
                    + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT'
                    + --count number of tables requested (number of commas plus one)

                    '
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')'
                    + --Count of the tables requested how many exist in the db

                    '
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
```

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_Grn-InvoiceDetails

```
Box.dbo.udf_SplitString(@ListOfTables,'',')) '
                    + --only if the count matches (all the tables exist in the
requested db) then run the script

                    '
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
                Insert #GrnMatching
                    ( DatabaseName
                    , Supplier
                    , Grn
                    , TransactionType
                    , Journal
                    , EntryNumber
                    , Invoice
                    , MatchedValue
                    , MatchedDate
                    , MatchedYear
                    , MatchedMonth
                    , MatchedQty
                    )
                Select DatabaseName = @DBCode
                    , Supplier
                    , Grn
                    , TransactionType
                    , Journal
                    , EntryNumber
                    , Invoice
                    , MatchedValue
                    , MatchedDate
                    , MatchedYear
                    , MatchedMonth
                    , MatchedQty
                    From GrnMatching
            End
    End';
                Declare @SQL1b Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end'
                    + --Only query DBs beginning SysProCompany

                    '
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN'
                    + --only companies selected in main run, or if companies
selected then all

                    '
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
                    + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT'
                    + --count number of tables requested (number of commas plus one)

                    '
```

Author: Johnson, Chris

```
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')'
                    + --Count of the tables requested how many exist in the db

                    '
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'','')) '
                    + --only if the count matches (all the tables exist in the
requested db) then run the script

                    '
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
            Insert #GrnDetails
                    ( DatabaseName, PurchaseOrder, PurchaseOrderLin, DebitRecGlCode,
StockCode, QtyReceived, Supplier, Grn, GrnSource, Journal, JournalEntry, [Orig-
ReceiptDate] )
            SELECT DatabaseName = @DBCode
                 , PurchaseOrder
                 , PurchaseOrderLin
                 , DebitRecGlCode
                 , StockCode
                 , QtyReceived
                 , Supplier
                 , Grn
                 , GrnSource
                 , Journal
                 , JournalEntry
                 , [OrigReceiptDate]  FROM GrnDetails
            End
    End';
                Declare @SQL2 Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end'
                    + --Only query DBs beginning SysProCompany

                    '
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN'
                    + --only companies selected in main run, or if companies
selected then all

                    '
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
                    + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT'
                    + --count number of tables requested (number of commas plus one)

                    '
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')'
                    + --Count of the tables requested how many exist in the db

                    '
```

Author: Johnson, Chris

```
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'','')) '
                    + --only if the count matches (all the tables exist in the
requested db) then run the script

                    '
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
                Insert #PorMasterDetail
                    ( DatabaseName
                    , MRequisition
                    , MRequisitionLine
                    , PurchaseOrder
                    , Line
                    )
                SELECT DatabaseName = @DBCode
                     , MRequisition
                     , MRequisitionLine
                     , PurchaseOrder
                     , Line FROM PorMasterDetail

                Insert #ReqRouting
                    ( DatabaseName
                    , Requisition
                    , RequisitionLine
                    , Operator
                    )
                SELECT DatabaseName = @DBCode
                 , Requisition
                 , RequisitionLine
                 , Operator FROM ReqRouting
                 where Operator  <> ''''
            End
    End';
                Declare @SQLReq Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end'
                    + --Only query DBs beginning SysProCompany

                    '
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN'
                    + --only companies selected in main run, or if companies
selected then all

                    '
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
                    + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT'
                    + --count number of tables requested (number of commas plus one)

                    '
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
```

```
String](@ListOfTables,'','')'
                    + --Count of the tables requested how many exist in the db

                    '
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'','')) '
                    + --only if the count matches (all the tables exist in the
requested db) then run the script

                    '
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
        Insert  [#ReqDetails] ( [DatabaseName], [Requisition], [Line], [UserCode])
        Select  DatabaseName = @DBCode
        , [Requisition]
        , [Line]
        , [UserCode]
        From    [ReqDetail];
            End
    End';


--Enable this function to check script changes (try to run script directly against
db manually)

--Print @SQL1a

--Print @SQL1b

--Print @SQL2


--execute script against each db, populating the base tables

                Print 'A';
                Exec [Process].[ExecForEachDB] @cmd = @SQL1a;
                Print 'B';
                Exec [Process].[ExecForEachDB] @cmd = @SQL1b;
                Print 'C';
                Exec [Process].[ExecForEachDB] @cmd = @SQL2;
                Print 'D';
                Exec [Process].[ExecForEachDB] @cmd = @SQLReq;
                Print 'Run';
--define the results you want to return


--Placeholder to create indexes as required


--script to combine base data and insert into results table

                Insert  [Report].[GrnInvoiceDetails]
                        ( [DatabaseName]
                        , [Supplier]
                        , [Grn]
                        , [TransactionType]
                        , [Journal]
```

Author: Johnson, Chris

```
                        , [EntryNumber]
                        , [Invoice]
                        , [PurchaseOrder]
                        , [PurchaseOrderLine]
                        , [Requisition]
                        , [RequisitionLine]
                        , [GlCode]
                        , [Description]
                        , [MatchedValue]
                        , [MatchedDate]
                        , [StockCode]
                        , [QtyReceived]
                        , [MatchedYear]
                        , [MatchedMonth]
                        , [MatchedQty]
                        , [Operator]
                        , [Approver]
                        , [OrigReceiptDate]
                        , [LoadDate]
                        )
                        Select  @Company
                              , [GM].[Supplier]
                              , [GM].[Grn]
                              , [GM].[TransactionType]
                              , [GM].[Journal]
                              , [GM].[EntryNumber]
                              , [GM].[Invoice]
                              , [GD].[PurchaseOrder]
                              , [GD].[PurchaseOrderLin]
                              , [PD].[MRequisition]
                              , [PD].[MRequisitionLine]
                              , [GD].[DebitRecGlCode]
                              , [GMS].[Description]
                              , [GM].[MatchedValue]
                              , [GM].[MatchedDate]
                              , [GD].[StockCode]
                              , [GD].[QtyReceived]
                              , [GM].[MatchedYear]
                              , [GM].[MatchedMonth]
                              , [GM].[MatchedQty]
                              , [RR].[Operator]
                              , [Approver] = [RD].[UserCode]
                              , [GD].[OrigReceiptDate]
                              , @LoadDate
                        From    [#GrnMatching] [GM]
                                Inner Join [#GrnDetails] [GD] On [GM].[Supplier] =
[GD].[Supplier]
                                                                 And [GM].[Grn] =
[GD].[Grn]
                                                                 And [GM].[Transaction-
Type] = [GD].[GrnSource]
                                                                 And [GM].[Journal] =
[GD].[Journal]
                                                                 And [GM].[EntryNumber]
= [GD].[JournalEntry]
                                                                 And [GD].[Database-
Name] = [GM].[DatabaseName]
                                Inner Join [SysproCompany40].[dbo].[GenMaster] [GMS]
On [GMS].[GlCode] = [GD].[DebitRecGlCode] Collate Latin1_General_BIN
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_Grn-InvoiceDetails

```
                                Inner Join [#PorMasterDetail] [PD] On [GD].[Purchase-
Order] = [PD].[PurchaseOrder]
                                                                   And [GD].[Purchase-
OrderLin] = [PD].[Line]
                                                                   And [PD].[Database-
Name] = [GD].[DatabaseName]
                                Left Outer Join [#ReqRouting] [RR] On
[RR].[Requisition] = [PD].[MRequisition]
                                                                   And [RR].[Requisition-
Line] = [PD].[MRequisitionLine]
                                                                   And [RR].[Database-
Name] = [PD].[DatabaseName]
                                Left Join [#ReqDetails] As [RD] On
[RD].[Requisition] = [RR].[Requisition]
                                                                   And [RD].[Line] =
[RR].[RequisitionLine]
                                                                   And [RD].[Database-
Name] = [RR].[DatabaseName];

                Update  [Process].[GrnInvoiceDetails]
                Set     [IsComplete] = 1
                Where   [LoadDate] = @LoadDate
                        And [Company] = @Company;

                Set @IsComplete = 1;
            End;


        While @IsComplete < 1
            Begin
                WaitFor Delay '00:00:01';
                Select  @IsComplete = [GID].[IsComplete]
                From    [Process].[GrnInvoiceDetails] As [GID]
                Where   [GID].[Company] = @Company
                        And [GID].[LoadDate] = @LoadDate;
            End;


--return results

        Select  [Company] = [GID].[DatabaseName]
              , [GID].[Supplier]
              , [GID].[Grn]
              , [GID].[TransactionType]
              , [GID].[Journal]
              , [GID].[EntryNumber]
              , [GID].[Invoice]
              , [GID].[PurchaseOrder]
              , [GID].[PurchaseOrderLine]
              , [GID].[Requisition]
              , [GID].[RequisitionLine]
              , [GID].[GlCode]
              , [GID].[Description]
              , [GID].[MatchedValue]
              , [GID].[MatchedDate]
              , [GID].[StockCode]
              , [GID].[QtyReceived]
              , [GID].[MatchedYear]
              , [GID].[MatchedMonth]
```

Author: Johnson, Chris

```
              , [GID].[MatchedQty]
              , [GID].[Operator]
              , [GID].[Approver]
              , [GID].[OrigReceiptDate]
        From    [Report].[GrnInvoiceDetails] As [GID]
        Where   [GID].[LoadDate] = @LoadDate
                And [GID].[DatabaseName] = @Company;

    End;

GO
EXEC sp_addextendedproperty N'MS_Description', N'provides details of goods received
in relation to invoices', 'SCHEMA', N'Report', 'PROCEDURE', N'UspResults_GrnInvoice-
Details', NULL, NULL
GO
```

Uses

[Process].[GrnInvoiceDetails]
[Report].[GrnInvoiceDetails]
[Process].[ExecForEachDB]
[Process].[UspInsert_RedTagLogs]
[Report]

[Report].[UspResults_InventoryInInspection]

MS_Description

list of inventory in inspection

Parameters

Name	Data Type	Max Length (Bytes)
@Company	varchar(max)	max
@RedTagType	char	1
@RedTagUse	varchar(500)	500

SQL Script

```
CREATE Proc [Report].[UspResults_InventoryInInspection]
    (
      @Company Varchar(Max)
    , @RedTagType Char(1)
    , @RedTagUse Varchar(500)
    )
As
    Begin
/*
Template designed by Chris Johnson, Prometic Group September 2015
Stored procedure set out to query multiple databases with the same information and
return it in a collated format
*/
        If IsNumeric(@Company) = 0
            Begin
                Select  @Company = Upper(@Company);
            End;

--remove nocount on to speed up query

        Set NoCount On;

--Red tag

        Declare @RedTagDB Varchar(255)= Db_Name();
        Exec [Process].[UspInsert_RedTagLogs] @StoredProcDb = 'BlackBox' ,
            @StoredProcSchema = 'Report' ,
            @StoredProcName = 'UspResults_InventoryInInspection' ,
            @UsedByType = @RedTagType , @UsedByName = @RedTagUse ,
            @UsedByDb = @RedTagDB;

--list the tables that are to be pulled back from each DB - if they are not found
the script will not be run against that db

        Declare @ListOfTables Varchar(Max) = 'InvInspect,InvMaster,PorMasterDetail';

--create temporary tables to be pulled from different databases, including a column
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-InventoryInInspection

```
to id

        Create Table [#InvInspect]
            (
              [DatabaseName] Varchar(150) Collate Latin1_General_BIN
            , [Grn] Varchar(20) Collate Latin1_General_BIN
            , [Lot] Varchar(50) Collate Latin1_General_BIN
            , [InspNarration] Varchar(100) Collate Latin1_General_BIN
            , [StockCode] Varchar(30) Collate Latin1_General_BIN
            , [PurchaseOrder] Varchar(20) Collate Latin1_General_BIN
            , [QtyAdvised] Numeric(20 , 8)
            , [QtyInspected] Numeric(20 , 8)
            , [QtyRejected] Numeric(20 , 8)
            , [InspectCompleted] Char(1) Collate Latin1_General_BIN
            , [DeliveryDate] Date
            , [PurchaseOrderLin] Int
            , [GrnReceiptDate] Date
            );
        Create Table [#PorMasterDetail]
            (
              [DatabaseName] Varchar(150) Collate Latin1_General_BIN
            , [MStockingUom] Varchar(10) Collate Latin1_General_BIN
            , [PurchaseOrder] Varchar(20) Collate Latin1_General_BIN
            , [Line] Int
            );
        Create Table [#InvMaster]
            (
              [DatabaseName] Varchar(150) Collate Latin1_General_BIN
            , [StockCode] Varchar(30) Collate Latin1_General_BIN
            , [Description] Varchar(50) Collate Latin1_General_BIN
            );


--create script to pull data from each db into the tables

        Declare @SQLInvInspect Varchar(Max) = 'USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            BEGIN
                Insert [#InvInspect]
                        ( [DatabaseName]
                        , [Grn]
                        , [Lot]
                        , [InspNarration]
                        , [StockCode]
                        , [PurchaseOrder]
                        , [QtyAdvised]
                        , [QtyInspected]
                        , [QtyRejected]
                        , [InspectCompleted]
                        , [DeliveryDate]
                        , [PurchaseOrderLin]
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-InventoryInInspection

```
                        , [GrnReceiptDate]
                        )
                SELECT [DatabaseName]=@DBCode
                     , [II].[Grn]
                     , [II].[Lot]
                     , [II].[InspNarration]
                     , [II].[StockCode]
                     , [II].[PurchaseOrder]
                     , [II].[QtyAdvised]
                     , [II].[QtyInspected]
                     , [II].[QtyRejected]
                     , [II].[InspectCompleted]
                     , [II].[DeliveryDate]
                     , [II].[PurchaseOrderLin]
                     , [II].[GrnReceiptDate]
                FROM [InvInspect] As [II]
            End
    End';
        Declare @SQLPorMasterDetail Varchar(Max) = 'USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            BEGIN
                Insert [#PorMasterDetail]
                        ( [DatabaseName]
                        , [MStockingUom]
                        , [PurchaseOrder]
                        , [Line]
                        )
                SELECT [DatabaseName]=@DBCode
                     , [PMD].[MStockingUom]
                     , [PMD].[PurchaseOrder]
                     , [PMD].[Line]
                FROM [PorMasterDetail] As [PMD]
            End
    End';
        Declare @SQLInvMaster Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            BEGIN
                Insert [#InvMaster]
                        ( [DatabaseName]
                        , [StockCode]
                        , [Description]
                        )
                SELECT [DatabaseName]=@DBCode
                     , [IM].[StockCode]
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-InventoryInInspection

```
                    , [IM].[Description] FROM [InvMaster] As [IM]
            End
    End';


--Enable this function to check script changes (try to run script directly against
db manually)


--execute script against each db, populating the base tables

        Exec [Process].[ExecForEachDB_WithTableCheck] @cmd = @SQLInvInspect ,
            @SchemaTablesToCheck = @ListOfTables;

        Exec [Process].[ExecForEachDB_WithTableCheck] @cmd = @SQLPorMasterDetail , -
- nvarchar(max)

            @SchemaTablesToCheck = @ListOfTables;

        Exec [Process].[ExecForEachDB_WithTableCheck] @cmd = @SQLInvMaster , --
nvarchar(max)

            @SchemaTablesToCheck = @ListOfTables;


--define the results you want to return

        Create Table [#Results]
            (
              [DatabaseName] Varchar(150) Collate Latin1_General_BIN
            , [Grn] Varchar(20) Collate Latin1_General_BIN
            , [Lot] Varchar(50) Collate Latin1_General_BIN
            , [InspNarration] Varchar(100) Collate Latin1_General_BIN
            , [StockCode] Varchar(30) Collate Latin1_General_BIN
            , [PurchaseOrder] Varchar(20) Collate Latin1_General_BIN
            , [QtyAdvised] Numeric(20 , 8)
            , [QtyInspected] Numeric(20 , 8)
            , [QtyRejected] Numeric(20 , 8)
            , [MStockingUom] Varchar(10) Collate Latin1_General_BIN
            , [InspectCompleted] Char(1) Collate Latin1_General_BIN
            , [DeliveryDate] Date
            , [StockDescription] Varchar(50) Collate Latin1_General_BIN
            , [ReceiptDate] Date
            );

--Placeholder to create indexes as required


--script to combine base data and insert into results table

        Insert  [#Results]
                ( [DatabaseName]
                , [Grn]
                , [Lot]
                , [InspNarration]
                , [StockCode]
                , [PurchaseOrder]
                , [QtyAdvised]
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-InventoryInInspection

```
                , [QtyInspected]
                , [QtyRejected]
                , [MStockingUom]
                , [InspectCompleted]
                , [DeliveryDate]
                , [StockDescription]
                , [ReceiptDate]
                )
                Select  [II].[DatabaseName]
                      , [II].[Grn]
                      , [II].[Lot]
                      , [II].[InspNarration]
                      , [II].[StockCode]
                      , [II].[PurchaseOrder]
                      , [II].[QtyAdvised]
                      , [II].[QtyInspected]
                      , [II].[QtyRejected]
                      , [PMD].[MStockingUom]
                      , [II].[InspectCompleted]
                      , [II].[DeliveryDate]
                      , [IM].[Description]
                      , [ReceiptDate] = [II].[GrnReceiptDate]
                From    [#InvInspect] As [II]
                        Left  Join [#PorMasterDetail] As [PMD]
                            On [PMD].[PurchaseOrder] = [II].[PurchaseOrder]
                               And [PMD].[Line] = [II].[PurchaseOrderLin]
                        Left Join [#InvMaster] As [IM]
                            On [IM].[StockCode] = [II].[StockCode]
                               And [IM].[DatabaseName] = [II].[DatabaseName]
                Where   Coalesce([II].[InspectCompleted] , 'N') <> 'Y';

--return results

        Select  [R].[DatabaseName]
              , [CN].[CompanyName]
              , [Grn] = Case When IsNumeric([R].[Grn]) = 1
                             Then Convert(Varchar(20) , Convert(Int , [R].[Grn]))
                             Else [R].[Grn]
                        End
              , [Lot] = Case When IsNumeric([R].[Lot]) = 1
                             Then Convert(Varchar(20) , Convert(Int , [R].[Lot]))
                             Else [R].[Lot]
                        End
              , [R].[InspNarration]
              , [R].[StockCode]
              , [PurchaseOrder] = Case When IsNumeric([R].[PurchaseOrder]) = 1
                                       Then Convert(Varchar(20) , Convert(Int ,
[R].[PurchaseOrder]))
                                       Else [R].[PurchaseOrder]
                                  End
              , [R].[QtyAdvised]
              , [R].[QtyInspected]
              , [R].[QtyRejected]
              , [MOrderUom] = [R].[MStockingUom]
              , [R].[InspectCompleted]
              , [R].[DeliveryDate]
              , [R].[StockDescription]
```

Author: Johnson, Chris

```
              , [R].[ReceiptDate]
        From    [#Results] [R]
                Left Join [Lookups].[CompanyNames] As [CN]
                    On [R].[DatabaseName] = [CN].[Company];

    End;

GO
EXEC sp_addextendedproperty N'MS_Description', N'list of inventory in inspection',
'SCHEMA', N'Report', 'PROCEDURE', N'UspResults_InventoryInInspection', NULL, NULL
GO
```

Uses

[Lookups].[CompanyNames]
[Process].[ExecForEachDB_WithTableCheck]
[Process].[UspInsert_RedTagLogs]
[Report]

[Report].[UspResults_InventoryInspectionTimes]

MS_Description

list of inventory in inspection with completion times

Parameters

Name	Data Type	Max Length (Bytes)
@Company	varchar(max)	max
@RedTagType	char	1
@RedTagUse	varchar(500)	500

SQL Script

```
CREATE Proc [Report].[UspResults_InventoryInspectionTimes]
    (
      @Company Varchar(Max)
    , @RedTagType Char(1)
    , @RedTagUse Varchar(500)
    )
As
    Begin
/*
Template designed by Chris Johnson, Prometic Group September 2015
Stored procedure set out to query multiple databases with the same information and
return it in a collated format
*/
        If IsNumeric(@Company) = 0
            Begin
                Select  @Company = Upper(@Company);
            End;

--remove nocount on to speed up query

        Set NoCount On;

--Red tag

        Declare @RedTagDB Varchar(255)= Db_Name();
        Exec [Process].[UspInsert_RedTagLogs] @StoredProcDb = 'BlackBox' ,
            @StoredProcSchema = 'Report' ,
            @StoredProcName = 'UspResults_InventoryInspectionTimes' ,
            @UsedByType = @RedTagType , @UsedByName = @RedTagUse ,
            @UsedByDb = @RedTagDB;

--list the tables that are to be pulled back from each DB - if they are not found
the script will not be run against that db

        Declare @ListOfTables Varchar(Max) = 'GrnDetails,InvInspect';

--create temporary tables to be pulled from different databases, including a column
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-
InventoryInspectionTimes

```
to id

        Create Table [#PorMasterDetail]
            (
              [DB] Varchar(150)                    collate latin1_general_bin
            , [PurchaseOrder] Varchar(20)         collate latin1_general_bin
            , [Line] Int
            , [MPrice] Numeric(20 , 2)
            );
        Create Table [#InvInspect]
            (
              [DB] Varchar(150)                    collate latin1_general_bin
            , [Lot] Varchar(50)                    collate latin1_general_bin
            , [Grn] Varchar(20)                    collate latin1_general_bin
            , [DeliveryDate] Date
            , [GrnReceiptDate] Date
            , [InspNarration] Varchar(100)         collate latin1_general_bin
            , [ExpiryDate] Date
            , [StockCode] Varchar(30)             collate latin1_general_bin
            , [QtyAdvised] Numeric(20 , 8)
            , [QtyInspected] Numeric(20 , 8)
            , [QtyAccepted] Numeric(20 , 8)
            , [QtyScrapped] Numeric(20 , 8)
            , [QtyRejected] Numeric(20 , 8)
            , [InspectCompleted] Char(1)          collate latin1_general_bin
            , [Supplier] Varchar(15)              collate latin1_general_bin
            , [PurchaseOrder] Varchar(20)         collate latin1_general_bin
            , [PurchaseOrderLin] Int
            , [SupDelNote] Varchar(50)             collate latin1_general_bin
            );
        Create Table [#InvInspectDet]
            (
              [DB] Varchar(150)                    collate latin1_general_bin
            , [TrnDate] Date
            , [Grn] Varchar(20)                    collate latin1_general_bin
            , [Lot] Varchar(50)                    collate latin1_general_bin
            , [TrnType] Char(1)                    collate latin1_general_bin
            );
        Create Table [#InvMaster]
            (
              [DB] Varchar(150)                    collate latin1_general_bin
            , [StockCode] Varchar(30)             collate latin1_general_bin
            , [InspectionFlag] Char(1)             collate latin1_general_bin
            , [TraceableType] Char(1)             collate latin1_general_bin
            , [Description] Varchar(50)            collate latin1_general_bin
            , [StockUom] Varchar(10)              collate latin1_general_bin
            );




--create script to pull data from each db into the tables

        Declare @SQLGrnDetails Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-InventoryInspectionTimes

```
Name(),len(db_Name())-13) else null end'
            + --Only query DBs beginning SysProCompany

            '
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN'
            + --only companies selected in main run, or if companies selected then
all

            '
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT'
            + --count number of tables requested (number of commas plus one)

            '
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')'
            + --Count of the tables requested how many exist in the db

            '
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'','')) '
            + --only if the count matches (all the tables exist in the requested db)
then run the script

            '
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
                Insert [#PorMasterDetail]
                    ( [DB]
                    , [PurchaseOrder]
                    , [Line]
                    , [MPrice]
                    )
            SELECT [DB]=@DBCode
                 , [PMD].[PurchaseOrder]
                 , [PMD].[Line]
                 , [PMD].[MPrice] FROM [PorMasterDetail] As [PMD]
            End
    End';
        Declare @SQLInvInspect Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end'
            + --Only query DBs beginning SysProCompany

            '
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN'
            + --only companies selected in main run, or if companies selected then
all

            '
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-InventoryInspectionTimes

```
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT'
            + --count number of tables requested (number of commas plus one)

            '
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')'
            + --Count of the tables requested how many exist in the db

            '
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'','')) '
            + --only if the count matches (all the tables exist in the requested db)
then run the script

            '
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
                Insert [#InvInspect]
                    ( [DB]
                    , [Lot]
                    , [Grn]
                    , [DeliveryDate]
                    , [GrnReceiptDate]
                    , [InspNarration]
                    , [ExpiryDate]
                    , [StockCode]
                    , [QtyAdvised]
                    , [QtyInspected]
                    , [QtyAccepted]
                    , [QtyScrapped]
                    , [QtyRejected]
                    , [InspectCompleted]
                    , [Supplier]
                    , [PurchaseOrder]
                    , [PurchaseOrderLin]
                    , [SupDelNote]
                    )
                SELECT [DB]=@DBCode
                     , [II].[Lot]
                  , [II].[Grn]
                  , [II].[DeliveryDate]
                  , [II].[GrnReceiptDate]
                  , [II].[InspNarration]
                  , [II].[ExpiryDate]
                  , [II].[StockCode]
                  , [II].[QtyAdvised]
                  , [II].[QtyInspected]
                  , [II].[QtyAccepted]
                  , [II].[QtyScrapped]
                  , [II].[QtyRejected]
                  , [II].[InspectCompleted]
                  , [II].[Supplier]
                  , [II].[PurchaseOrder]
```

Author: Johnson, Chris

```
                  , [II].[PurchaseOrderLin]
                  , [II].[SupDelNote] FROM [InvInspect] As [II]
            End
    End';
        Declare @SQLInvMaster Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end'
            + --Only query DBs beginning SysProCompany

            '
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN'
            + --only companies selected in main run, or if companies selected then
all

            '
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables + '''
                  , @RequiredCountOfTables INT
                  , @ActualCountOfTables INT'
            + --count number of tables requested (number of commas plus one)

            '
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')'
            + --Count of the tables requested how many exist in the db

            '
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'','')) '
            + --only if the count matches (all the tables exist in the requested db)
then run the script

            '
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
                Insert [#InvMaster]
                        ( [DB]
                        , [StockCode]
                        , [InspectionFlag]
                        , [TraceableType]
                        )
                SELECT [DB]=@DBCode
                     , [IM].[StockCode]
                     , [IM].[InspectionFlag]
                     , [TraceableType]
                FROM [InvMaster] As [IM]
            End
    End';
        Declare @SQLInvInspectDet Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end'
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-InventoryInspectionTimes

```
            + --Only query DBs beginning SysProCompany

            '
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN'
            + --only companies selected in main run, or if companies selected then
all

            '
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT'
            + --count number of tables requested (number of commas plus one)

            '
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')'
            + --Count of the tables requested how many exist in the db

            '
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'','')) '
            + --only if the count matches (all the tables exist in the requested db)
then run the script

            '
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
                Insert [#InvInspectDet]
                        ( [DB]
                        , [TrnDate]
                        , [Grn]
                        , [Lot]
                        , [TrnType]
                        )
                SELECT [DB]=@DBCode
                     , [IID].[TrnDate]
                     , [IID].[Grn]
                     , [IID].[Lot]
                     , [IID].[TrnType] FROM [InvInspectDet] As [IID]
            End
    End';
--Enable this function to check script changes (try to run script directly against
db manually)

--Print @SQL


--execute script against each db, populating the base tables

        Exec [Process].[ExecForEachDB] @cmd = @SQLGrnDetails;
        Exec [Process].[ExecForEachDB] @cmd = @SQLInvInspect;
        Exec [Process].[ExecForEachDB] @cmd = @SQLInvInspectDet;
        Exec [Process].[ExecForEachDB] @cmd = @SQLInvMaster;
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-InventoryInspectionTimes

```
--define the results you want to return

        Create Table [#Results]
            (
              [Grn] Varchar(20)                    collate latin1_general_bin
            , [Lot] Varchar(50)                    collate latin1_general_bin
            , [Supplier] Varchar(15)             collate latin1_general_bin
            , [OrigReceiptDate] Date
            , [PurchaseOrder] Varchar(20)         collate latin1_general_bin
            , [PurchaseOrderLine] Int
            , [StockCode] Varchar(30)             collate latin1_general_bin
            , [StockDescription] Varchar(50)     collate latin1_general_bin
            , [SupCatalogueNum] Varchar(50)        collate latin1_general_bin
            , [QtyUom] Varchar(10)                collate latin1_general_bin
            , [DeliveryDate] Date
            , [GrnReceiptDate] Date
            , [SupplierLot] Varchar(100)         collate latin1_general_bin
            , [ExpiryDate] Date
            , [QtyAdvised] Numeric(20 , 8)
            , [QtyInspected] Numeric(20 , 8)
            , [QtyAccepted] Numeric(20 , 8)
            , [QtyScrapped] Numeric(20 , 8)
            , [QtyRejected] Numeric(20 , 8)
            , [InspectCompleted] Char(1)         collate latin1_general_bin
            , [TrnDate] Date
            , [WorkingDaysToApprove] Int
            , [DatabaseName] Varchar(150)         collate latin1_general_bin
            , [Price] Numeric(20 , 2)
            );

--Placeholder to create indexes as required


--script to combine base data and insert into results table

        Insert  [#Results]
                ( [Grn]
                , [Lot]
                , [Supplier]
                , [OrigReceiptDate]
                , [PurchaseOrder]
                , [PurchaseOrderLine]
                , [StockCode]
                , [StockDescription]
                , [SupCatalogueNum]
                , [QtyUom]
                , [DeliveryDate]
                , [GrnReceiptDate]
                , [SupplierLot]
                , [ExpiryDate]
                , [QtyAdvised]
                , [QtyInspected]
                , [QtyAccepted]
                , [QtyScrapped]
                , [QtyRejected]
                , [InspectCompleted]
                , [TrnDate]
```

Author: Johnson, Chris

```
                , [WorkingDaysToApprove]
                , [DatabaseName]
                , [Price]
                )
                Select  [II].[Grn]
                      , [II].[Lot]
                      , [II].[Supplier]
                      , [II].[DeliveryDate]
                      , [II].[PurchaseOrder]
                      , [II].[PurchaseOrderLin]
                      , [II].[StockCode]
                      , [IM].[Description]
                      , [II].[SupDelNote]
                      , [IM].[StockUom]
                      , [II].[DeliveryDate]
                      , [II].[GrnReceiptDate]
                      , [SupplierLot] = [II].[InspNarration]
                      , [II].[ExpiryDate]
                      , [II].[QtyAdvised]
                      , [II].[QtyInspected]
                      , [II].[QtyAccepted]
                      , [II].[QtyScrapped]
                      , [II].[QtyRejected]
                      , [InspectCompleted] = Case When Coalesce([II].[Inspect-
Completed] ,
                                                                '') = ''
                                                 Then 'N'
                                                 Else [II].[InspectCompleted]
                                            End
                      , [IID].[TrnDate]
                      , [WorkingDaysToApprove] = [BlackBox].[Process].[Udf_Working-
Days]([II].[DeliveryDate] ,
                                                                Coalesce([IID].[Trn-
Date] ,
                                                                GetDate()) ,
                                                                'UK')
                      , [II].[DB]
                      , [PMD].[MPrice]
                From    [#InvInspect] As [II]
                        Left Join [#InvInspectDet] As [IID] On [IID].[Grn] =
[II].[Grn]
                                                               And [IID].[Lot] =
[II].[Lot]
                                                               And [IID].[TrnType] =
'A'
                                                               And [IID].[DB] =
[II].[DB]
                        Left Join [#InvMaster] As [IM] On [IM].[StockCode] =
[II].[StockCode]
                                                          And [IM].[DB] = [II].[DB]
                        Left Join [#PorMasterDetail] As [PMD] On [PMD].[Purchase-
Order] = [II].[PurchaseOrder]
                                                               And [PMD].[Line] =
[II].[PurchaseOrderLin]
                                                               And [PMD].[DB] =
[II].[DB]
                Where   [IM].[InspectionFlag] = 'Y'
                        And [IM].[TraceableType] = 'T';
--return results
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-InventoryInspectionTimes

```
        Select  [CN].[CompanyName]
              , [R].[Grn]
              , [R].[Lot]
              , [R].[Supplier]
              , [R].[OrigReceiptDate]
              , [R].[PurchaseOrder]
              , [R].[PurchaseOrderLine]
              , [R].[StockCode]
              , [R].[StockDescription]
              , [R].[SupCatalogueNum]
              , [R].[Price]
              , [R].[QtyUom]
              , [R].[DeliveryDate]
              , [R].[GrnReceiptDate]
              , [R].[SupplierLot]
              , [R].[ExpiryDate]
              , [R].[QtyAdvised]
              , [R].[QtyInspected]
              , [R].[QtyAccepted]
              , [R].[QtyScrapped]
              , [R].[QtyRejected]
              , [R].[InspectCompleted]
              , [CompletedDate] = [R].[TrnDate]
              , [R].[WorkingDaysToApprove]
              , [ReportStatus] = Case When [R].[InspectCompleted] = 'N'
                                      Then 'Still awaiting inspection'
                                      When [R].[WorkingDaysToApprove] <= 1
                                      Then '1 day or less'
                                      When [R].[WorkingDaysToApprove] <= 5
                                      Then '5 days or less'
                                      When [R].[WorkingDaysToApprove] <= 10
                                      Then '10 days or less'
                                      When [R].[WorkingDaysToApprove] <= 15
                                      Then '15 days or less'
                                      Else 'More than 15 days'
                                 End
              , [DeliveryYear] = Year([R].[DeliveryDate])
              , [DeliveryMonth] = Month([R].[DeliveryDate])
              , [DeliveryQuarter] = Ceiling(Month([R].[DeliveryDate]) / 3.0)
        From    [#Results] [R]
                Left Join [Lookups].[CompanyNames] As [CN] On [CN].[Company] =
[R].[DatabaseName];


    End;

GO
EXEC sp_addextendedproperty N'MS_Description', N'list of inventory in inspection
with completion times', 'SCHEMA', N'Report', 'PROCEDURE', N'UspResults_Inventory-
InspectionTimes', NULL, NULL
GO
```

Uses

[Lookups].[CompanyNames]
[Process].[ExecForEachDB]

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-InventoryInspectionTimes

[Process].[UspInsert_RedTagLogs]

[Process].[Udf_WorkingDays]

[Report]

Author: Johnson, Chris

[Report].[UspResults_InvoiceRunVerify]

MS_Description

not in use

Parameters

Name	Data Type	Max Length (Bytes)
@Company	varchar(max)	max
@RedTagType	char	1
@RedTagUse	varchar(500)	500

SQL Script

```
CREATE Proc [Report].[UspResults_InvoiceRunVerify]
    (
      @Company Varchar(Max)
    , @RedTagType Char(1)
    , @RedTagUse Varchar(500)
    )
As
    Begin
/*
Template designed by Chris Johnson, Prometic Group September 2015
Stored procedure set out to query multiple databases with the same information and
return it in a collated format
*/
        If IsNumeric(@Company) = 0
            Begin
                Select  @Company = Upper(@Company);
            End;

--remove nocount on to speed up query

        Set NoCount On;

--Red tag

        Declare @RedTagDB Varchar(255)= Db_Name();
        Exec [Process].[UspInsert_RedTagLogs] @StoredProcDb = 'BlackBox' ,
            @StoredProcSchema = 'Report' ,
            @StoredProcName = 'UspResults_InvoiceRunVerify' ,
            @UsedByType = @RedTagType , @UsedByName = @RedTagUse ,
            @UsedByDb = @RedTagDB;

--list the tables that are to be pulled back from each DB - if they are not found
the script will not be run against that db

        Declare @ListOfTables Varchar(Max) = 'AssetDepreciation,TblApTerms';

--create temporary tables to be pulled from different databases, including a column
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-InvoiceRunVerify

```
to id

        Create Table [#ApSupplier]
            (
              [DatabaseName] Varchar(150)    collate latin1_general_bin
            , [Supplier] Varchar(50)        collate latin1_general_bin
            , [SupplierName] Varchar(255)    collate latin1_general_bin
            , [Currency] Varchar(5)           collate latin1_general_bin
            );
        Create Table [#ApInvoice]
            (
              [DatabaseName] Varchar(150)    collate latin1_general_bin
            , [Supplier] Varchar(50)        collate latin1_general_bin
            , [Invoice] Varchar(50)           collate latin1_general_bin
            , [EntryNumber] Int
            , [Journal] Int
            , [PaymentNumber] Int
            , [ConvRate] Float
            , [MulDiv] Varchar(5)            collate latin1_general_bin
            , [MthInvBal1] Numeric(20 , 2)
            , [InvoiceYear] Int
            , [InvoiceMonth] Int
            , [JournalDate] Date
            , [Bank] Varchar(10)           collate latin1_general_bin
            , [InvoiceDate] Date
            , [DueDate] Date
            , [OrigInvValue] Numeric(20 , 2)
            );
        Create Table [#ApJnlSummary]
            (
              [DatabaseName] Varchar(150)    collate latin1_general_bin
            , [Invoice] Varchar(50)           collate latin1_general_bin
            , [EntryNumber] Int
            , [Journal] Int
            , [TransactionCode] Varchar(5)    collate latin1_general_bin
            , [InvoiceValue] Numeric(20 , 2)
            );
        Create Table [#ApJnlDistrib]
            (
              [DatabaseName] Varchar(150)    collate latin1_general_bin
            , [Journal] Int
            , [EntryNumber] Int
            , [DistrValue] Numeric(20 , 2)
            , [ExpenseGlCode] Varchar(50)    collate latin1_general_bin
            , [ExpenseType] Varchar(5)        collate latin1_general_bin
            );
        Create Table [#ApPayRunDet]
            (
              [DatabaseName] Varchar(150)    collate latin1_general_bin
            , [Invoice] Varchar(50)           collate latin1_general_bin
            , [PaymentNumber] Int
            , [Supplier] Varchar(50)        collate latin1_general_bin
            , [Cheque] Varchar(30)           collate latin1_general_bin
            , [PostCurrency] Varchar(10)    collate latin1_general_bin
            , [ChequeDate] Date
            , [PostConvRate] Float
            , [ChRegister] Int
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-InvoiceRunVerify

```
            , [PostMulDiv] Varchar(5)        collate latin1_general_bin
            );
        Create Table [#ApPayRunHdr]
            (
              [DatabaseName] Varchar(150)    collate latin1_general_bin
            , [PaymentNumber] Int
            , [PaymentDate] Date
            , [PayYear] Int
            , [PayMonth] Int
            , [Operator] Varchar(250)        collate latin1_general_bin
            );

--create script to pull data from each db into the tables

        Declare @SQLApSupplier Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end'
            + --Only query DBs beginning SysProCompany

            '
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN'
            + --only companies selected in main run, or if companies selected then
all

            '
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT'
            + --count number of tables requested (number of commas plus one)

            '
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')'
            + --Count of the tables requested how many exist in the db

            '
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'',''))  '
            + --only if the count matches (all the tables exist in the requested db)
then run the script

            '
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
                Insert [#ApSupplier]
                        ( [DatabaseName]
                        , [Supplier]
                        , [SupplierName]
                        , [Currency]
                        )
                SELECT [DatabaseName]=@DBCode
                     , [AS].[Supplier]
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-InvoiceRunVerify

```
                    , [AS].[SupplierName]
                    , [AS].[Currency]
                     FROM [ApSupplier] As [AS]
            End
    End';
        Declare @SQLApInvoice Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end'
            + --Only query DBs beginning SysProCompany

            '
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN'
            + --only companies selected in main run, or if companies selected then
all

            '
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT'
            + --count number of tables requested (number of commas plus one)

            '
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')'
            + --Count of the tables requested how many exist in the db

            '
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'','')) '
            + --only if the count matches (all the tables exist in the requested db)
then run the script

            '
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
                Insert [#ApInvoice]
                        ( [DatabaseName]
                        , [Supplier]
                        , [Invoice]
                        , [EntryNumber]
                        , [Journal]
                        , [PaymentNumber]
                        , [ConvRate]
                        , [MulDiv]
                        , [MthInvBal1]
                        , [InvoiceYear]
                        , [InvoiceMonth]
                        , [JournalDate]
                        , [Bank]
                        , [InvoiceDate]
                        , [DueDate]
                        , [OrigInvValue]
```

Author: Johnson, Chris

```
                        )
                SELECT [DatabaseName]=@DBCode
                     , [AI].[Supplier]
                     , [AI].[Invoice]
                     , [AI].[EntryNumber]
                     , [AI].[Journal]
                     , [AI].[PaymentNumber]
                     , [AI].[ConvRate]
                     , [AI].[MulDiv]
                     , [AI].[MthInvBal1]
                     , [AI].[InvoiceYear]
                     , [AI].[InvoiceMonth]
                     , [AI].[JournalDate]
                     , [AI].[Bank]
                     , [AI].[InvoiceDate]
                     , [AI].[DueDate]
                     , [AI].[OrigInvValue] FROM [ApInvoice] As [AI]
            End
    End';
        Declare @SQLApJnlSummary Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end'
            + --Only query DBs beginning SysProCompany

            '
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN'
            + --only companies selected in main run, or if companies selected then
all

            '
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT'
            + --count number of tables requested (number of commas plus one)

            '
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')'
            + --Count of the tables requested how many exist in the db

            '
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'','')) '
            + --only if the count matches (all the tables exist in the requested db)
then run the script

            '
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
                Insert [#ApJnlSummary]
                        ( [DatabaseName]
                        , [Invoice]
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-InvoiceRunVerify

```
                        , [EntryNumber]
                        , [Journal]
                        , [TransactionCode]
                        , [InvoiceValue]
                        )
                SELECT [DatabaseName]=@DBCode
                     , [AJS].[Invoice]
                     , [AJS].[EntryNumber]
                     , [AJS].[Journal]
                     , [AJS].[TransactionCode]
                     , [AJS].[InvoiceValue] FROM [ApJnlSummary] As [AJS]
            End
    End';
        Declare @SQLApJnlDistrib Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end'
            + --Only query DBs beginning SysProCompany

            '
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN'
            + --only companies selected in main run, or if companies selected then
all

            '
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT'
            + --count number of tables requested (number of commas plus one)

            '
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')'
            + --Count of the tables requested how many exist in the db

            '
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'','')) '
            + --only if the count matches (all the tables exist in the requested db)
then run the script

            '
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
                Insert [#ApJnlDistrib]
                        ( [DatabaseName]
                        , [Journal]
                        , [EntryNumber]
                        , [DistrValue]
                        , [ExpenseGlCode]
                        , [ExpenseType]
                        )
                SELECT [DatabaseName]=@DBCode
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-InvoiceRunVerify

```
                    , [AJD].[Journal]
                    , [AJD].[EntryNumber]
                    , [AJD].[DistrValue]
                    , [AJD].[ExpenseGlCode]
                    , [AJD].[ExpenseType] FROM [ApJnlDistrib] As [AJD]
            End
    End';
        Declare @SQLApPayRunDet Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end'
            + --Only query DBs beginning SysProCompany

            '
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN'
            + --only companies selected in main run, or if companies selected then
all

            '
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT'
            + --count number of tables requested (number of commas plus one)

            '
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')'
            + --Count of the tables requested how many exist in the db

            '
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'','')) '
            + --only if the count matches (all the tables exist in the requested db)
then run the script

            '
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
                Insert [#ApPayRunDet]
                        ( [DatabaseName]
                        , [Invoice]
                        , [PaymentNumber]
                        , [Supplier]
                        , [Cheque]
                        , [PostCurrency]
                        , [ChequeDate]
                        , [PostConvRate]
                        , [ChRegister]
                        , [PostMulDiv]
                        )
                SELECT [DatabaseName]=@DBCode
                     , [APRD].[Invoice]
                     , [APRD].[PaymentNumber]
```

Author: Johnson, Chris

```
                    , [APRD].[Supplier]
                    , [APRD].[Cheque]
                    , [APRD].[PostCurrency]
                    , [APRD].[ChequeDate]
                    , [APRD].[PostConvRate]
                    , [APRD].[ChRegister]
                    , [APRD].[PostMulDiv] FROM [ApPayRunDet] As [APRD]
            End
    End';
        Declare @SQLApPayRunHdr Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end'
            + --Only query DBs beginning SysProCompany

            '
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN'
            + --only companies selected in main run, or if companies selected then
all

            '
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT'
            + --count number of tables requested (number of commas plus one)

            '
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')'
            + --Count of the tables requested how many exist in the db

            '
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'','')) '
            + --only if the count matches (all the tables exist in the requested db)
then run the script

            '
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
                Insert [#ApPayRunHdr]
                        ( [DatabaseName]
                        , [PaymentNumber]
                        , [PaymentDate]
                        , [PayYear]
                        , [PayMonth]
                        , [Operator]
                        )
                SELECT [DatabaseName]=@DBCode
                     , [APRH].[PaymentNumber]
                     , [APRH].[PaymentDate]
                     , [APRH].[PayYear]
                     , [APRH].[PayMonth]
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-InvoiceRunVerify

```
                    , [APRH].[Operator] FROM [ApPayRunHdr] As [APRH]
            End
    End';
--Enable this function to check script changes (try to run script directly against
db manually)

--Print @SQL


--execute script against each db, populating the base tables

        Exec [Process].[ExecForEachDB] @cmd = @SQLApSupplier;
        Exec [Process].[ExecForEachDB] @cmd = @SQLApInvoice;
        Exec [Process].[ExecForEachDB] @cmd = @SQLApJnlSummary;
        Exec [Process].[ExecForEachDB] @cmd = @SQLApJnlDistrib;
        Exec [Process].[ExecForEachDB] @cmd = @SQLApPayRunDet;
        Exec [Process].[ExecForEachDB] @cmd = @SQLApPayRunHdr;

--define the results you want to return

        Create Table [#Results]
            (
              [DatabaseName] Varchar(150)            collate latin1_general_bin
            , [Supplier] Varchar(15)                collate latin1_general_bin
            , [Invoice] Varchar(20)                   collate latin1_general_bin
            , [PostCurrency] Char(3)                collate latin1_general_bin
            , [ConvRate] Float
            , [MulDiv] Char(1)                        collate latin1_general_bin
            , [MthInvBal] Float
            , [CompLocalAmt] Decimal(10 , 2)
            , [DistrValue] Float
            , [Description] Varchar(50)               collate latin1_general_bin
            , [ExpenseGlCode] Varchar(35)           collate latin1_general_bin
            , [InvoiceYear] Int
            , [InvoiceMonth] Int
            , [JournalDate] Date
            , [InvoiceDate] Date
            , [PaymentNumber] Int
            , [Bank] Varchar(15)                    collate latin1_general_bin
            , [PaymentDate] Date
            , [PayYear] Int
            , [PayMonth] Int
            , [Operator] Varchar(20)                collate latin1_general_bin
            , [ChRegister] Float
            , [Cheque] Varchar(15)                    collate latin1_general_bin
            , [ChequeDate] Date
            , [InvNetPayValue] Decimal(15 , 3)
            , [DueDate] Date
            , [InvoiceType] Char(1)                   collate latin1_general_bin
            , [PostValue] Decimal(15 , 3)
            , [PostConvRate] Float
            , [PostMulDiv] Char(1)                    collate latin1_general_bin
            , [SupplierName] Varchar(50)            collate latin1_general_bin
            , [ExpenseType] Varchar(150)            collate latin1_general_bin
            , [CompanyName] Varchar(300)            collate latin1_general_bin
            , [SupplierCurrency] Varchar(5)           collate latin1_general_bin
            );
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-InvoiceRunVerify

```
--Placeholder to create indexes as required


--script to combine base data and insert into results table

        Insert  [#Results]
                ( [DatabaseName]
                , [Supplier]
                , [Invoice]
                , [PostCurrency]
                , [ConvRate]
                , [MulDiv]
                , [MthInvBal]
                , [CompLocalAmt]
                , [DistrValue]
                , [Description]
                , [ExpenseGlCode]
                , [InvoiceYear]
                , [InvoiceMonth]
                , [JournalDate]
                , [InvoiceDate]
                , [PaymentNumber]
                , [Bank]
                , [PaymentDate]
                , [PayYear]
                , [PayMonth]
                , [Operator]
                , [ChRegister]
                , [Cheque]
                , [ChequeDate]
                , [InvNetPayValue]
                , [DueDate]
                , [InvoiceType]
                , [PostValue]
                , [PostConvRate]
                , [PostMulDiv]
                , [SupplierName]
                , [ExpenseType]
                , [CompanyName]
                , [SupplierCurrency]
                )
                Select  [Company] = [AI].[DatabaseName]
                      , [Supplier] = [AS].[Supplier]
                      , [Invoice] = [AI].[Invoice]
                      , [PostCurrency] = [AD].[PostCurrency]
                      , [ConvRate] = [AI].[ConvRate]
                      , [MulDiv] = [AI].[MulDiv]
                      , [MthInvBal] = [AI].[MthInvBal1]
                      , [CompLocalAmt] = Cast(Case When [AI].[MulDiv] = 'M'
                                                   Then [AI].[MthInvBal1]
                                                        * [AI].[ConvRate]
                                                   Else [AI].[MthInvBal1]
                                                        / [AI].[ConvRate]
                                              End As Numeric(20 , 2))
                      , [DistrValue] = [AJD].[DistrValue]
                      , [Description] = [GM].[Description]
```

Author: Johnson, Chris

```
                        , [ExpenseGlCode] = Case When [AJD].[ExpenseGlCode] = ''
                                                 Then Null
                                                 Else [AJD].[ExpenseGlCode]
                                            End
                        , [InvoiceYear] = [AI].[InvoiceYear]
                        , [InvoiceMonth] = [AI].[InvoiceMonth]
                        , [JournalDate] = [AI].[JournalDate]
                        , [InvoiceDate] = [AI].[InvoiceDate]
                        , [PaymentNumber] = [AI].[PaymentNumber]
                        , [Bank] = [AI].[Bank]
                        , [PaymentDate] = [APH].[PaymentDate]
                        , [PayYear] = [APH].[PayYear]
                        , [PayMonth] = [APH].[PayMonth]
                        , [Operator] = [APH].[Operator]
                        , [ChRegister] = [AD].[ChRegister]
                        , [Cheque] = [AD].[Cheque]
                        , [ChequeDate] = [AD].[ChequeDate]
                        , [InvNetPayValue] = [AJS].[InvoiceValue]
                        , [DueDate] = [AI].[DueDate]
                        , [InvoiceType] = [AJS].[TransactionCode]--[AJS].[InvoiceType]

                        , [PostValue] = [AI].[OrigInvValue]
                        , [PostConvRate] = [AD].[PostConvRate]
                        , [PostMulDiv] = [AD].[PostMulDiv]
                        , [SupplierName] = [AS].[SupplierName]
                        , [ExpenseType] = [AJDE].[ExpenseTypeDesc]
                        , [CompanyName] = [CN].[CompanyName]
                        , [SupplierCurrency] = [AS].[Currency]
                    From    [#ApSupplier] As [AS]
                            Left Join [#ApInvoice] As [AI] On [AI].[Supplier] =
[AS].[Supplier]
                                                              And [AI].[DatabaseName] =
[AS].[DatabaseName]
                            Left Join [#ApJnlSummary] As [AJS] On [AJS].[Invoice] =
[AI].[Invoice]
                                                                  And [AJS].[Entry-
Number] = [AI].[EntryNumber]
                                                                  And [AJS].[Journal] =
[AI].[Journal]
                                                                  And [AJS].[Database-
Name] = [AI].[DatabaseName]
                            Left Join [#ApJnlDistrib] As [AJD] On [AJD].[Journal] =
[AI].[Journal]
                                                                  And [AJD].[Entry-
Number] = [AI].[EntryNumber]
                                                                  And [AJD].[Database-
Name] = [AI].[DatabaseName]
                            Left Join [BlackBox].[Lookups].[ApJnlDistribExpenseType]
[AJDE] On [AJD].[ExpenseType] = [AJDE].[ExpenseType]
                            Left Join [SysproCompany40].[dbo].[GenMaster] As [GM] On
[AJD].[ExpenseGlCode] = [GM].[GlCode]
                                                                  And [AJD].[Database-
Name] = [GM].[Company]
                            Left Join [#ApPayRunDet] [AD] On [AD].[Invoice] =
[AI].[Invoice]
                                                             And [AD].[PaymentNumber] =
[AI].[PaymentNumber]
                                                             And [AD].[Supplier] =
[AI].[Supplier]
                                                             And [AD].[DatabaseName] =
[AI].[DatabaseName]
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-InvoiceRunVerify

```
                        Left Join [#ApPayRunHdr] [APH] On [APH].[PaymentNumber] =
[AI].[PaymentNumber]
                                                          And [APH].[DatabaseName] =
[AI].[DatabaseName]
                        Left Join [BlackBox].[Lookups].[CcompanyNames] As [CN] On
[CN].[Company] = [AI].[DatabaseName]
                Where   [AI].[Invoice] Is Not Null;


--return results

        Select  [CompanyName] = [DatabaseName]
              , [Supplier]
              , [Invoice]
              , [PostCurrency]
              , [ConvRate]
              , [MulDiv]
              , [MthInvBal]
              , [CompLocalAmt]
              , [DistrValue]
              , [Description]
              , [ExpenseGlCode]
              , [InvoiceYear]
              , [InvoiceMonth]
              , [JournalDate]
              , [InvoiceDate]
              , [PaymentNumber]
              , [Bank]
              , [PaymentDate]
              , [PayYear]
              , [PayMonth]
              , [Operator]
              , [ChRegister]
              , [Cheque]
              , [ChequeDate]
              , [InvNetPayValue]
              , [DueDate]
              , [InvoiceType]
              , [PostValue]
              , [PostConvRate]
              , [PostMulDiv]
              , [SupplierName]
              , [ExpenseType]
              , [CompanyName]
              , [SupplierCurrency]
        From    [#Results];

    End;

GO
EXEC sp_addextendedproperty N'MS_Description', N'not in use', 'SCHEMA', N'Report',
'PROCEDURE', N'UspResults_InvoiceRunVerify', NULL, NULL
GO
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-InvoiceRunVerify

Uses

[Lookups].[ApJnlDistribExpenseType]

[Lookups].[CompanyNames]

[Process].[ExecForEachDB]

[Process].[UspInsert_RedTagLogs]

[Report]

Author: Johnson, Chris

[Report].[UspResults_JobHeader]

MS_Description

high level details of Job

Parameters

Name	Data Type	Max Length (Bytes)
@Company	varchar(max)	max
@Job	varchar(150)	150
@RedTagType	char	1
@RedTagUse	varchar(500)	500

SQL Script

```
CREATE Proc [Report].[UspResults_JobHeader]
    (
      @Company Varchar(Max)
    , @Job Varchar(150)
    , @RedTagType Char(1)
    , @RedTagUse Varchar(500)
    )
As
    Begin
/*
Template designed by Chris Johnson, Prometic Group September 2015
Stored procedure set out to query multiple databases with the same information and
return it in a collated format
--exec Report.UspResults_JobHeader  10, 'FA1408'
*/
        If IsNumeric(@Company) = 0
            Begin
                Select  @Company = Upper(@Company);
            End;

--remove nocount on to speed up query

        Set NoCount On;

--Red tag

        Declare @RedTagDB Varchar(255)= Db_Name();
        Exec [Process].[UspInsert_RedTagLogs] @StoredProcDb = 'BlackBox' ,
            @StoredProcSchema = 'Report' ,
            @StoredProcName = 'UspResults_JobHeader' ,
            @UsedByType = @RedTagType , @UsedByName = @RedTagUse ,
            @UsedByDb = @RedTagDB;

        Select  @Job = Case When IsNumeric(@Job) = 1
                            Then Right('000000000000000' + @Job , 15)
```

Author: Johnson, Chris

```
                            Else @Job
                        End;

--list the tables that are to be pulled back from each DB - if they are not found
the script will not be run against that db

        Declare @ListOfTables Varchar(Max) = 'WipMaster,InvMaster';

--create temporary tables to be pulled from different databases, including a column
to id

        Create Table [#WipMaster]
            (
              [DatabaseName] Varchar(150) Collate Latin1_General_BIN
            , [Job] Varchar(35) Collate Latin1_General_BIN
            , [JobDescription] Varchar(150) Collate Latin1_General_BIN
            , [JobClassification] Varchar(10) Collate Latin1_General_BIN
            , [ProducedStockCode] Varchar(35) Collate Latin1_General_BIN
            , [ProducedStockDescription] Varchar(150)
                Collate Latin1_General_BIN
            , [UomFlag] Char(1) Collate Latin1_General_BIN
            , [JobTenderDate] DateTime2
            , [JobDeliveryDate] DateTime2
            , [JobStartDate] DateTime2
            , [ActCompleteDate] DateTime2
            , [Complete] Char(1) Collate Latin1_General_BIN
            , [QtyManufactured] Float
            , [SalesOrder] Varchar(35) Collate Latin1_General_BIN
            , [SellingPrice] Float
            , [QtyToMake] Numeric(20 , 8)
            );
        Create Table [#InvMaster]
            (
              [DatabaseName] Varchar(150) Collate Latin1_General_BIN
            , [StockCode] Varchar(35) Collate Latin1_General_BIN
            , [StockUom] Varchar(10) Collate Latin1_General_BIN
            , [CostUom] Varchar(10) Collate Latin1_General_BIN
            , [OtherUom] Varchar(10) Collate Latin1_General_BIN
            , [AlternateUom] Varchar(10) Collate Latin1_General_BIN
            , [IssMultLotsFlag] Char(1) Collate Latin1_General_BIN
            );
        Create Table [#InvMovementsSummary]
            (
              [DatabaseName] Varchar(150) Collate Latin1_General_BIN
            , [Job] Varchar(50) Collate Latin1_General_BIN
            , [MaterialValue] Numeric(20 , 7)
            );
        Create Table [#WipLabJnlSummary]
            (
              [DatabaseName] Varchar(500) Collate Latin1_General_BIN
            , [Job] Varchar(50) Collate Latin1_General_BIN
            , [LabourValue] Numeric(20 , 7)
            , [TimeTotal] Numeric(20 , 7)
            );
        Create Table [#LotsHeader]
            (
              [DatabaseName] Varchar(150) Collate Latin1_General_BIN
```

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_Job-Header

```
            , [JobPurchOrder] Varchar(50) Collate Latin1_General_BIN
            , [Lot] Varchar(50) Collate Latin1_General_BIN
            );
        Create Table [#SalesSummary]
            (
              [DatabaseName] Varchar(150) Collate Latin1_General_BIN
            , [Job] Varchar(50) Collate Latin1_General_BIN
            , [SellingValue] Numeric(20 , 7)
            , [RemovedValue] Numeric(20 , 7)
            );

        Set @Job = Upper(@Job);


--create script to pull data from each db into the tables

        Declare @SQL1 Varchar(Max) = 'USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end'
            + --Only query DBs beginning SysProCompany

            '
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN'
            + --only companies selected in main run, or if companies selected then
all

            '
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT'
            + --count number of tables requested (number of commas plus one)

            '
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')'
            + --Count of the tables requested how many exist in the db

            '
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'','')) '
            + --only if the count matches (all the tables exist in the requested db)
then run the script

            '
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
                Insert #WipMaster
                        ( DatabaseName
                        , Job
                        , JobDescription
                        , JobClassification
                        , ProducedStockCode
                        , ProducedStockDescription
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_Job-Header

```
                        , UomFlag
                        , JobTenderDate
                        , JobDeliveryDate
                        , JobStartDate
                        , ActCompleteDate
                        , Complete
                        , QtyManufactured
                        , SalesOrder
                        , SellingPrice
                        , QtyToMake
                        )
                SELECT DatabaseName =@DBCode
                     , Job
                     , JobDescription
                     , JobClassification
                     , StockCode
                     , StockDescription
                     , UomFlag
                     , JobTenderDate
                     , JobDeliveryDate
                     , JobStartDate
                     , ActCompleteDate
                     , Complete
                     , QtyManufactured
                     , SalesOrder
                     , SellingPrice
                     , QtyToMake
                      FROM WipMaster
                Where upper(Job) =''' + @Job + '''
            End
    End';
        Declare @SQL2 Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end'
            + --Only query DBs beginning SysProCompany

            '
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN'
            + --only companies selected in main run, or if companies selected then
all

            '
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT'
            + --count number of tables requested (number of commas plus one)

            '
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')'
            + --Count of the tables requested how many exist in the db

            '
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_Job-Header

```
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'',')) '
            + --only if the count matches (all the tables exist in the requested db)
then run the script

            '
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
            Insert #InvMaster
                ( DatabaseName
                , StockCode
                , StockUom
                , CostUom
                , OtherUom
                , AlternateUom
                , IssMultLotsFlag
                )
            SELECT DatabaseName =@DBCode
             , StockCode
             , StockUom
             , CostUom
             , OtherUom
             , AlternateUom
             , IssMultLotsFlag FROM InvMaster
            End
    End';
        Declare @SQL3 Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end'
            + --Only query DBs beginning SysProCompany

            '
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN'
            + --only companies selected in main run, or if companies selected then
all

            '
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT'
            + --count number of tables requested (number of commas plus one)

            '
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'',')'
            + --Count of the tables requested how many exist in the db

            '
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'',')) '
            + --only if the count matches (all the tables exist in the requested db)
then run the script
```

Author: Johnson, Chris

```
            '
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
            Insert  [#InvMovementsSummary]
                    ( [DatabaseName]
                    , [Job]
                    , [MaterialValue]
                    )
            Select
                DatabaseName = @DBCode
              , IM.Job
              , MaterialValue = SUM(IM.TrnValue * TT.AmountModifier)
            From
                InvMovements IM
            Left Join BlackBox.Lookups.TrnTypeAmountModifier TT
                On TT.TrnType = IM.TrnType Collate Latin1_General_BIN
                   And TT.Company = @DBCode
            Where
                [IM].[TrnType] <> ''R''
                And [IM].[Job] = ''' + @Job + '''
            Group By
                IM.Job
            End
    End';
        Declare @SQL4 Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end'
            + --Only query DBs beginning SysProCompany

            '
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN'
            + --only companies selected in main run, or if companies selected then
all

            '
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT'
            + --count number of tables requested (number of commas plus one)

            '
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')'
            + --Count of the tables requested how many exist in the db

            '
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'',''))  '
            + --only if the count matches (all the tables exist in the requested db)
then run the script
```

Author: Johnson, Chris

```
            '
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
            Insert  [#WipLabJnlSummary]
                    ( [DatabaseName]
                    , [Job]
                    , [LabourValue]
                    , TimeTotal
                    )
                    Select
                        @DBCode
                      , Job
                      , LabourValue = SUM(LabourValue)
                      , TimeTotal = SUM( coalesce([RunTime]        ,0)
                                        +coalesce([SetUpTime]    ,0)
                                        +coalesce([StartUpTime]    ,0)
                                        +coalesce([TeardownTime],0)
                                       )
                    From
                        WipLabJnl
                    Where
                        [Job] = ''' + @Job + '''
                    Group By
                    Job;
            End
    End';
        Declare @SQL5 Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end'
            + --Only query DBs beginning SysProCompany

            '
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN'
            + --only companies selected in main run, or if companies selected then
all

            '
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT'
            + --count number of tables requested (number of commas plus one)

            '
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')'
            + --Count of the tables requested how many exist in the db

            '
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'','')) '
            + --only if the count matches (all the tables exist in the requested db)
then run the script
```

Author: Johnson, Chris

```
            '
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
            Insert  [#LotsHeader]
        ( [JobPurchOrder]
        , [Lot]
        , DatabaseName
        )
        Select Distinct
            [l].[JobPurchOrder]
          , [l].[Lot]
          , @DBCode
        From
            [LotTransactions] As [l]
        Left Join [dbo].[InvMaster] As [im]
            On [im].[StockCode] = [l].[StockCode]
        Where
            [JobPurchOrder] = ''' + @Job + ''';
            End
    End';
        Declare @SQL6 Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end'
            + --Only query DBs beginning SysProCompany

            '
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN'
            + --only companies selected in main run, or if companies selected then
all

            '
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT'
            + --count number of tables requested (number of commas plus one)

            '
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')'
            + --Count of the tables requested how many exist in the db

            '
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'',''))  '
            + --only if the count matches (all the tables exist in the requested db)
then run the script

            '
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
            Insert  [#SalesSummary]
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_Job-Header

```
        ( [DatabaseName]
        , [Job]
        , [SellingValue]
        , RemovedValue
        )
        Select
            @DBCode
          , Job = [l].[JobPurchOrder]
          , [SellingValue] = SUM([wm].[SellingPrice])
          , RemovedValue = SUM(lt.[TrnQuantity])
        From
            LotTransactions As [lt]
        Inner Join [#LotsHeader] As [l]
            On [l].[Lot] = [lt].[Lot] Collate Latin1_General_BIN
               And [l].[DatabaseName] = @DBCode
        Left Join WipMaster As [wm]
            On [wm].[Job] = [lt].[Job] Collate Latin1_General_BIN
        Where
            [wm].[SellingPrice] <> 0
            And [l].[JobPurchOrder] = ''' + @Job + '''
        Group By
            [l].[JobPurchOrder];
            End
    End';
--Enable this function to check script changes (try to run script directly against db manually)

--Print @SQL


--execute script against each db, populating the base tables

        Exec [Process].[ExecForEachDB] @cmd = @SQL1;
        Exec [Process].[ExecForEachDB] @cmd = @SQL2;
        Exec [Process].[ExecForEachDB] @cmd = @SQL3;
        Exec [Process].[ExecForEachDB] @cmd = @SQL4;
        Exec [Process].[ExecForEachDB] @cmd = @SQL5;
        Exec [Process].[ExecForEachDB] @cmd = @SQL6;

--define the results you want to return

        Create Table [#Results]
            (
              [DatabaseName] Varchar(150) Collate Latin1_General_BIN
            , [Job] Varchar(50) Collate Latin1_General_BIN
            , [JobDescription] Varchar(150) Collate Latin1_General_BIN
            , [JobClassification] Varchar(50) Collate Latin1_General_BIN
            , [ProducedStockCode] Varchar(50) Collate Latin1_General_BIN
            , [ProducedStockDescription] Varchar(150) Collate Latin1_General_BIN
            , [ProducedQty] Numeric(20 , 7)
            , [Uom] Varchar(10) Collate Latin1_General_BIN
            , [JobTenderDate] DateTime2
            , [JobDeliveryDate] DateTime2
            , [JobStartDate] DateTime2
            , [ActCompleteDate] DateTime2
            , [Complete] Char(1) Collate Latin1_General_BIN
            , [QtyManufactured] Numeric(20 , 7)
```

Author: Johnson, Chris

```
            , [SalesOrder] Varchar(50) Collate Latin1_General_BIN
            , [IssMultLotsFlag] Char(1) Collate Latin1_General_BIN
            , [SellingPrice] Numeric(20 , 7)
            , [MaterialValue] Numeric(20 , 7)
            , [LabourValue] Numeric(20 , 7)
            , [InputValue] Numeric(20 , 7)
            , [SellingValue] Numeric(20 , 7)
            , [Profit] Numeric(20 , 7)
            , [RemovedValue] Numeric(20 , 7)
            , [TimeTotal] Numeric(20 , 7)
            , [QtyToMake] Numeric(20 , 8)
            );

--Placeholder to create indexes as required

--create NonClustered Index Index_Name On #Table1 (DatabaseName) Include (Column-
Name)


--script to combine base data and insert into results table

        Insert  [#Results]
                ( [DatabaseName]
                , [Job]
                , [JobDescription]
                , [JobClassification]
                , [ProducedStockCode]
                , [ProducedStockDescription]
                , [ProducedQty]
                , [Uom]
                , [JobTenderDate]
                , [JobDeliveryDate]
                , [JobStartDate]
                , [ActCompleteDate]
                , [Complete]
                , [QtyManufactured]
                , [SalesOrder]
                , [IssMultLotsFlag]
                , [SellingPrice]
                , [MaterialValue]
                , [LabourValue]
                , [InputValue]
                , [SellingValue]
                , [Profit]
                , [RemovedValue]
                , [TimeTotal]
                , [QtyToMake]
                )
                Select  [WM].[DatabaseName]
                      , [WM].[Job]
                      , [WM].[JobDescription]
                      , [WM].[JobClassification]
                      , [ProducedStockCode] = [WM].[ProducedStockCode]
                      , [ProducedStockDescription] = [WM].[ProducedStockDescription]
                      , [ProducedQty] = [WM].[QtyManufactured]
                      , [Uom] = Case When [WM].[UomFlag] = 'S'
                                     Then [IM].[StockUom]
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_Job-Header

```
                                    When [WM].[UomFlag] = 'C'
                                    Then [IM].[CostUom]
                                    When [WM].[UomFlag] = 'O'
                                    Then [IM].[OtherUom]
                                    When [WM].[UomFlag] = 'A'
                                    Then [IM].[AlternateUom]
                               End
                     , [JobTenderDate] = [WM].[JobTenderDate]
                     , [JobDeliveryDate] = [WM].[JobDeliveryDate]
                     , [JobStartDate] = [WM].[JobStartDate]
                     , [ActCompleteDate] = [WM].[ActCompleteDate]
                     , [WM].[Complete]
                     , [WM].[QtyManufactured]
                     , [SalesOrder] = Case When [WM].[SalesOrder] = ''
                                           Then Null
                                           Else [WM].[SalesOrder]
                                      End
                     , [IM].[IssMultLotsFlag]
                     , [WM].[SellingPrice]
                     , [ims].[MaterialValue]
                     , [wljs].[LabourValue]
                     , [InputValue] = [ims].[MaterialValue]
                       + [wljs].[LabourValue]
                     , [ss].[SellingValue]
                     , [Profit] = [ss].[SellingValue] + [ims].[MaterialValue]
                       - [wljs].[LabourValue]
                     , [ss].[RemovedValue]
                     , [wljs].[TimeTotal]
                     , [WM].[QtyToMake]
                From    [#WipMaster] [WM]
                        Left Join [#InvMaster] [IM] On [IM].[StockCode] =
[WM].[ProducedStockCode]
                                                       And [IM].[DatabaseName] =
[WM].[DatabaseName]
                        Left Join [#InvMovementsSummary] As [ims] On [ims].[Database-
Name] = [WM].[DatabaseName]
                                                          And [ims].[Job] =
[WM].[Job]
                        Left Join [#WipLabJnlSummary] As [wljs] On [wljs].[Database-
Name] = [WM].[DatabaseName]
                                                          And [wljs].[Job] =
[WM].[Job]
                        Left Join [#SalesSummary] As [ss] On [ss].[Job] = [WM].[Job]
                                                          And [ss].[Job] =
[WM].[Job];

--return results

        Select  [CN].[CompanyName]
              , [r].[Job]
              , [r].[JobDescription]
              , [r].[JobClassification]
              , [r].[ProducedStockCode]
              , [r].[ProducedStockDescription]
              , [r].[ProducedQty]
              , [r].[Uom]
              , [JobTenderDate] = Cast([r].[JobTenderDate] As Date)
              , [JobDeliveryDate] = Cast([r].[JobDeliveryDate] As Date)
              , [JobStartDate] = Cast([r].[JobStartDate] As Date)
```

Author: Johnson, Chris

```
                , [ActCompleteDate] = Cast([r].[ActCompleteDate] As Date)
                , [r].[Complete]
                , [r].[QtyManufactured]
                , [r].[SalesOrder]
                , [r].[IssMultLotsFlag]
                , [r].[SellingPrice]
                , [MaterialValue] = [r].[MaterialValue] * -1
                , [r].[LabourValue]
                , [r].[InputValue]
                , [r].[SellingValue]
                , [r].[Profit]
                , [r].[RemovedValue]
                , [r].[TimeTotal]
                , [r].[QtyToMake]
        From    [#Results] As [r]
                Left Join [Lookups].[CompanyNames] [CN] On [CN].[Company] =
[r].[DatabaseName]  Collate Latin1_General_BIN;

        Drop Table [#Results];
    End;

GO
EXEC sp_addextendedproperty N'MS_Description', N'high level details of Job',
'SCHEMA', N'Report', 'PROCEDURE', N'UspResults_JobHeader', NULL, NULL
GO
```

Uses

[Lookups].[CompanyNames]
[Process].[ExecForEachDB]
[Process].[UspInsert_RedTagLogs]
[Report]

Author: Johnson, Chris

[Report].[UspResults_JobHeader_AllJobs]

MS_Description

high level details of Job, for all jobs

Parameters

Name	Data Type	Max Length (Bytes)
@Company	varchar(max)	max
@StartDateText	varchar(50)	50
@EndDateText	varchar(50)	50
@RedTagType	char	1
@RedTagUse	varchar(500)	500

SQL Script

```
CREATE Proc [Report].[UspResults_JobHeader_AllJobs]
    (
      @Company Varchar(Max)
    , @StartDateText Varchar(50)
    , @EndDateText Varchar(50)
    , @RedTagType Char(1)
    , @RedTagUse Varchar(500)
    )
As /*
Template designed by Chris Johnson, Prometic Group September 2015
*/
    Begin

        Declare @StartDate Date
          , @EndDate Date;
        Select  @StartDate = Cast(@StartDateText As Date);
        Select  @EndDate = Cast(@EndDateText As Date);

        If IsNumeric(@Company) = 0
            Begin
                Select  @Company = Upper(@Company);
            End;

--remove nocount on to speed up query

        Set NoCount On;

--Red tag

        Declare @RedTagDB Varchar(255)= Db_Name();
        Exec [Process].[UspInsert_RedTagLogs] @StoredProcDb = 'BlackBox' ,
            @StoredProcSchema = 'Report' ,
            @StoredProcName = 'UspResults_JobHeader_AllJobs' ,
            @UsedByType = @RedTagType , @UsedByName = @RedTagUse ,
```

Author: Johnson, Chris

```
            @UsedByDb = @RedTagDB;

--list the tables that are to be pulled back from each DB - if they are not found
the script will not be run against that db

        Declare @ListOfTables Varchar(Max) = 'WipMaster,InvMaster,InvMovements,Wip-
LabJnl,SorContractPrice';

--create temporary tables to be pulled from different databases, including a column
to id

        Create Table [#WipMaster]
            (
              [DatabaseName] Varchar(150) Collate Latin1_General_BIN
            , [Job] Varchar(35) Collate Latin1_General_BIN
            , [JobDescription] Varchar(150) Collate Latin1_General_BIN
            , [JobClassification] Varchar(10) Collate Latin1_General_BIN
            , [ProducedStockCode] Varchar(35) Collate Latin1_General_BIN
            , [ProducedStockDescription] Varchar(150)
                Collate Latin1_General_BIN
            , [UomFlag] Char(1) Collate Latin1_General_BIN
            , [JobTenderDate] DateTime2
            , [JobDeliveryDate] DateTime2
            , [JobStartDate] DateTime2
            , [ActCompleteDate] DateTime2
            , [Complete] Char(1) Collate Latin1_General_BIN
            , [QtyManufactured] Float
            , [SalesOrder] Varchar(35) Collate Latin1_General_BIN
            , [SellingPrice] Float
            );
        Create Table [#InvMaster]
            (
              [DatabaseName] Varchar(150) Collate Latin1_General_BIN
            , [StockCode] Varchar(35) Collate Latin1_General_BIN
            , [StockUom] Varchar(10) Collate Latin1_General_BIN
            , [CostUom] Varchar(10) Collate Latin1_General_BIN
            , [OtherUom] Varchar(10) Collate Latin1_General_BIN
            , [AlternateUom] Varchar(10) Collate Latin1_General_BIN
            , [IssMultLotsFlag] Char(1) Collate Latin1_General_BIN
            );
        Create Table [#InvMovementsSummary]
            (
              [DatabaseName] Varchar(150) Collate Latin1_General_BIN
            , [Job] Varchar(50) Collate Latin1_General_BIN
            , [MaterialValue] Numeric(20 , 7)
            );
        Create Table [#WipLabJnlSummary]
            (
              [DatabaseName] Varchar(500) Collate Latin1_General_BIN
            , [Job] Varchar(50) Collate Latin1_General_BIN
            , [LabourValue] Numeric(20 , 7)
            , [TimeTotal] Numeric(20 , 7)
            );
        Create Table [#LotsHeader]
            (
              [DatabaseName] Varchar(150) Collate Latin1_General_BIN
            , [JobPurchOrder] Varchar(50) Collate Latin1_General_BIN
            , [Lot] Varchar(50) Collate Latin1_General_BIN
```

Author: Johnson, Chris

```
            );
        Create Table [#SalesSummary]
            (
              [DatabaseName] Varchar(150) Collate Latin1_General_BIN
            , [Job] Varchar(50) Collate Latin1_General_BIN
            , [SellingValue] Numeric(20 , 7)
            , [RemovedValue] Numeric(20 , 7)
            );
        Create Table [#SorContractPrice]
            (
              [DatabaseName] Varchar(150) Collate Latin1_General_BIN
            , [StockCode] Varchar(50) Collate Latin1_General_BIN
            , [StartDate] DateTime2
            , [ExpiryDate] DateTime2
            , [MaxFixedPrice] Numeric(20 , 7)
            , [MinFixedPrice] Numeric(20 , 7)
            );

--create script to pull data from each db into the tables

        Declare @SQL1 Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end'
            + --Only query DBs beginning SysProCompany

            '
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN'
            + --only companies selected in main run, or if companies selected then
all

            '
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT'
            + --count number of tables requested (number of commas plus one)

            '
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')'
            + --Count of the tables requested how many exist in the db

            '
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'','')) '
            + --only if the count matches (all the tables exist in the requested db)
then run the script

            '
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
                Insert #WipMaster
                        ( DatabaseName
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_Job-Header_AllJobs

```
                        , Job
                        , JobDescription
                        , JobClassification
                        , ProducedStockCode
                        , ProducedStockDescription
                        , UomFlag
                        , JobTenderDate
                        , JobDeliveryDate
                        , JobStartDate
                        , ActCompleteDate
                        , Complete
                        , QtyManufactured
                        , SalesOrder
                        , SellingPrice
                        )
                SELECT DatabaseName =@DBCode
                     , Job
                     , JobDescription
                     , JobClassification
                     , StockCode
                     , StockDescription
                     , UomFlag
                     , JobTenderDate
                     , JobDeliveryDate
                     , JobStartDate
                     , ActCompleteDate
                     , Complete
                     , QtyManufactured
                     , SalesOrder
                     , SellingPrice
                      FROM WipMaster
                Where cast(JobStartDate as Date) between '''
            + Cast(@StartDate As Varchar(50)) + '''
                                                And '''
            + Cast(@EndDate As Varchar(50)) + '''
            End
    End';
        Declare @SQL2 Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end'
            + --Only query DBs beginning SysProCompany

            '
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN'
            + --only companies selected in main run, or if companies selected then
all

            '
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT'
            + --count number of tables requested (number of commas plus one)
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_Job-Header_AllJobs

```
            '
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')'
            + --Count of the tables requested how many exist in the db

            '
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'','')) '
            + --only if the count matches (all the tables exist in the requested db)
then run the script

            '
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
            Insert #InvMaster
                ( DatabaseName
                , StockCode
                , StockUom
                , CostUom
                , OtherUom
                , AlternateUom
                , IssMultLotsFlag
                )
            SELECT DatabaseName =@DBCode
             , StockCode
             , StockUom
             , CostUom
             , OtherUom
             , AlternateUom
             , IssMultLotsFlag FROM InvMaster
            End
    End';
        Declare @SQL3 Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end'
            + --Only query DBs beginning SysProCompany

            '
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN'
            + --only companies selected in main run, or if companies selected then
all

            '
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT'
            + --count number of tables requested (number of commas plus one)

            '
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')'
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_Job-Header_AllJobs

```
            + --Count of the tables requested how many exist in the db

            '
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'',')) '
            + --only if the count matches (all the tables exist in the requested db)
then run the script

            '
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
            Insert  [#InvMovementsSummary]
                    ( [DatabaseName]
                    , [Job]
                    , [MaterialValue]
                    )
            Select
                DatabaseName = @DBCode
              , IM.Job
              , MaterialValue = SUM(IM.TrnValue * TT.AmountModifier)
            From
                InvMovements IM
            Left Join BlackBox.Lookups.TrnTypeAmountModifier TT
                On TT.TrnType = IM.TrnType Collate Latin1_General_BIN
                   And TT.Company = @DBCode
            Inner Join #WipMaster W on IM.Job =W.Job
                    and W.DatabaseName = @DBCode
            Where
                [IM].[TrnType] <> ''R''
            Group By
                IM.Job
            End
    End';
        Declare @SQL4 Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end'
            + --Only query DBs beginning SysProCompany

            '
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN'
            + --only companies selected in main run, or if companies selected then
all

            '
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT'
            + --count number of tables requested (number of commas plus one)

            '
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')'
```

Author: Johnson, Chris

```
            + --Count of the tables requested how many exist in the db

            '
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'',')) '
            + --only if the count matches (all the tables exist in the requested db)
then run the script

            '
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
            Insert  [#WipLabJnlSummary]
                    ( [DatabaseName]
                    , [Job]
                    , [LabourValue]
                    , TimeTotal
                    )
                    Select
                        @DBCode
                      , WJL.Job
                      , LabourValue = SUM(WJL.LabourValue)
                      , TimeTotal = SUM( coalesce(WJL.[RunTime]        ,0)
                                        +coalesce(WJL.[SetUpTime]    ,0)
                                        +coalesce(WJL.[StartUpTime]    ,0)
                                        +coalesce(WJL.[TeardownTime],0)
                                       )
                    From
                        WipLabJnl WJL
                        inner join #WipMaster W on WJL.Job=W.Job
                        and W.DatabaseName=@DBCode
                    Group By
                    WJL.Job;
            End
    End';
        Declare @SQL5 Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end'
            + --Only query DBs beginning SysProCompany

            '
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN'
            + --only companies selected in main run, or if companies selected then
all

            '
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT'
            + --count number of tables requested (number of commas plus one)

            '
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
```

```
String](@ListOfTables,'','')'
            + --Count of the tables requested how many exist in the db

            '
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'','')) '
            + --only if the count matches (all the tables exist in the requested db)
then run the script

            '
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
            Insert  [#LotsHeader]
        ( [JobPurchOrder]
        , [Lot]
        , DatabaseName
        )
        Select Distinct
            [l].[JobPurchOrder]
          , [l].[Lot]
          , @DBCode
        From
            [LotTransactions] As [l]
        Left Join [dbo].[InvMaster] As [im]
            On [im].[StockCode] = [l].[StockCode]
        inner join #WipMaster w on w.Job=l.JobPurchOrder
            End
    End';
        Declare @SQL6 Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end'
            + --Only query DBs beginning SysProCompany

            '
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN'
            + --only companies selected in main run, or if companies selected then
all

            '
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT'
            + --count number of tables requested (number of commas plus one)

            '
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')'
            + --Count of the tables requested how many exist in the db

            '
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_Job-Header_AllJobs

```
Box.dbo.udf_SplitString(@ListOfTables,'',')) '
            + --only if the count matches (all the tables exist in the requested db) then run the script

            '
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
            Insert  [#SalesSummary]
        ( [DatabaseName]
        , [Job]
        , [SellingValue]
        , RemovedValue
        )
        Select
            @DBCode
          , Job = [l].[JobPurchOrder]
          , [SellingValue] = SUM([wm].[SellingPrice])
          , RemovedValue = SUM(lt.[TrnQuantity])
        From
            LotTransactions As [lt]
        Inner Join [#LotsHeader] As [l]
            On [l].[Lot] = [lt].[Lot] Collate Latin1_General_BIN
               And [l].[DatabaseName] = @DBCode
        Left Join WipMaster As [wm]
            On [wm].[Job] = [lt].[Job] Collate Latin1_General_BIN
        Inner Join #WipMaster As [w] on [l].[JobPurchOrder]=w.Job
                    and w.DatabaseName=@DBCode
        Where
            [wm].[SellingPrice] <> 0

        Group By
            [l].[JobPurchOrder];
            End
    End';
        Declare @SQL7 Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_Name(),len(db_Name())-13) else null end'
            + --Only query DBs beginning SysProCompany

            '
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN'
            + --only companies selected in main run, or if companies selected then all

            '
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT'
            + --count number of tables requested (number of commas plus one)

            '
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-String](@ListOfTables,'',')'
```

Author: Johnson, Chris

```
            + --Count of the tables requested how many exist in the db

            '
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'','')) '
            + --only if the count matches (all the tables exist in the requested db)
then run the script

            '
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
            Insert [#SorContractPrice]
                ( [DatabaseName]
                , [StockCode]
                , [StartDate]
                , [ExpiryDate]
                , [MaxFixedPrice]
                , [MinFixedPrice]
                )
            Select @DBCode
              , [StockCode]
              , [StartDate]
              , [ExpiryDate]
              , max([FixedPrice])
              , Min([FixedPrice])
            From SorContractPrice
            group by [StockCode]
              , [StartDate]
              , [ExpiryDate]
            End
    End';
--Enable this function to check script changes (try to run script directly against
db manually)

--Print @SQL7


--execute script against each db, populating the base tables

        --Print 1;

        Exec [Process].[ExecForEachDB] @cmd = @SQL1;
        --Print 2;

        Exec [Process].[ExecForEachDB] @cmd = @SQL2;
        --Print 3;

        Exec [Process].[ExecForEachDB] @cmd = @SQL3;
        --Print 4;

        Exec [Process].[ExecForEachDB] @cmd = @SQL4;
        --Print 5;

        Exec [Process].[ExecForEachDB] @cmd = @SQL5;
        --Print 6;

        Exec [Process].[ExecForEachDB] @cmd = @SQL6;
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_Job-Header_AllJobs

```
        --Print 7;

        Exec [Process].[ExecForEachDB] @cmd = @SQL7;
        --Print 8;

--define the results you want to return

        Create Table [#Results]
            (
              [DatabaseName] Varchar(150) Collate Latin1_General_BIN
            , [Job] Varchar(50) Collate Latin1_General_BIN
            , [JobDescription] Varchar(150) Collate Latin1_General_BIN
            , [JobClassification] Varchar(50) Collate Latin1_General_BIN
            , [ProducedStockCode] Varchar(50) Collate Latin1_General_BIN
            , [ProducedStockDescription] Varchar(150) Collate Latin1_General_BIN
            , [ProducedQty] Numeric(20 , 7)
            , [Uom] Varchar(10) Collate Latin1_General_BIN
            , [JobTenderDate] DateTime2
            , [JobDeliveryDate] DateTime2
            , [JobStartDate] DateTime2
            , [ActCompleteDate] DateTime2
            , [Complete] Char(1) Collate Latin1_General_BIN
            , [QtyManufactured] Numeric(20 , 7)
            , [SalesOrder] Varchar(50)
            , [IssMultLotsFlag] Char(1) Collate Latin1_General_BIN
            , [SellingPrice] Numeric(20 , 7)
            , [MaterialValue] Numeric(20 , 7)
            , [LabourValue] Numeric(20 , 7)
            , [InputValue] Numeric(20 , 7)
            , [SellingValue] Numeric(20 , 7)
            , [Profit] Numeric(20 , 7)
            , [RemovedValue] Numeric(20 , 7)
            , [TimeTotal] Numeric(20 , 7)
            , [MaxSellingPrice] Numeric(20 , 7)
            , [MinSellingPrice] Numeric(20 , 7)
            );

--Placeholder to create indexes as required

--create NonClustered Index Index_Name On #Table1 (DatabaseName) Include (Column-Name)

        --Print 9;

--script to combine base data and insert into results table

        Insert  [#Results]
                ( [DatabaseName]
                , [Job]
                , [JobDescription]
                , [JobClassification]
                , [ProducedStockCode]
                , [ProducedStockDescription]
                , [ProducedQty]
                , [Uom]
                , [JobTenderDate]
                , [JobDeliveryDate]
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_Job-Header_AllJobs

```
                , [JobStartDate]
                , [ActCompleteDate]
                , [Complete]
                , [QtyManufactured]
                , [SalesOrder]
                , [IssMultLotsFlag]
                , [SellingPrice]
                , [MaterialValue]
                , [LabourValue]
                , [InputValue]
                , [SellingValue]
                , [Profit]
                , [RemovedValue]
                , [TimeTotal]
                , [MaxSellingPrice]
                , [MinSellingPrice]
                )
                Select  [WM].[DatabaseName]
                      , [WM].[Job]
                      , [WM].[JobDescription]
                      , [WM].[JobClassification]
                      , [ProducedStockCode] = [WM].[ProducedStockCode]
                      , [ProducedStockDescription] = [WM].[ProducedStockDescription]
                      , [ProducedQty] = [WM].[QtyManufactured]
                      , [Uom] = Case When [WM].[UomFlag] = 'S'
                                     Then [IM].[StockUom]
                                     When [WM].[UomFlag] = 'C'
                                     Then [IM].[CostUom]
                                     When [WM].[UomFlag] = 'O'
                                     Then [IM].[OtherUom]
                                     When [WM].[UomFlag] = 'A'
                                     Then [IM].[AlternateUom]
                                End
                      , [JobTenderDate] = [WM].[JobTenderDate]
                      , [JobDeliveryDate] = [WM].[JobDeliveryDate]
                      , [JobStartDate] = [WM].[JobStartDate]
                      , [ActCompleteDate] = [WM].[ActCompleteDate]
                      , [WM].[Complete]
                      , [WM].[QtyManufactured]
                      , [SalesOrder] = Case When [WM].[SalesOrder] = ''
                                            Then Null
                                            Else [WM].[SalesOrder]
                                       End
                      , [IM].[IssMultLotsFlag]
                      , [WM].[SellingPrice]
                      , [ims].[MaterialValue]
                      , [wljs].[LabourValue]
                      , [InputValue] = Coalesce([ims].[MaterialValue] , 0)
                        - Coalesce([wljs].[LabourValue] , 0)
                      , [ss].[SellingValue]
                      , [Profit] = [ss].[SellingValue] + [ims].[MaterialValue]
                        - [wljs].[LabourValue]
                      , [ss].[RemovedValue]
                      , [wljs].[TimeTotal]
                      , [scp].[MaxFixedPrice]
                      , [scp].[MinFixedPrice]
                From    [#WipMaster] [WM]
```

Author: Johnson, Chris

```
                        Left Join [#InvMaster] [IM] On [IM].[StockCode] =
[WM].[ProducedStockCode]
                                                        And [IM].[DatabaseName] =
[WM].[DatabaseName]
                        Left Join [#InvMovementsSummary] As [ims] On [ims].[Database-
Name] = [WM].[DatabaseName]
                                                          And [ims].[Job] =
[WM].[Job]
                        Left Join [#WipLabJnlSummary] As [wljs] On [wljs].[Database-
Name] = [WM].[DatabaseName]
                                                          And [wljs].[Job] =
[WM].[Job]
                        Left Join [#SalesSummary] As [ss] On [ss].[Job] = [WM].[Job]
                                                        And [ss].[Job] =
[WM].[Job]
                        Left Join [#SorContractPrice] As [scp] On [scp].[StockCode]
= [IM].[StockCode]
                                                          And [scp].[Database-
Name] = [WM].[DatabaseName];
        --Print 10;

--return results

        Select  [CN].[CompanyName]
              , [r].[Job]
              , [r].[JobDescription]
              , [r].[JobClassification]
              , [r].[ProducedStockCode]
              , [r].[ProducedStockDescription]
              , [r].[ProducedQty]
              , [r].[Uom]
              , [JobTenderDate] = Cast([r].[JobTenderDate] As Date)
              , [JobDeliveryDate] = Cast([r].[JobDeliveryDate] As Date)
              , [JobStartDate] = Cast([r].[JobStartDate] As Date)
              , [ActCompleteDate] = Cast([r].[ActCompleteDate] As Date)
              , [Complete] = Case When [r].[Complete] = 'Y' Then 'Yes'
                                  Else 'No'
                             End
              , [r].[QtyManufactured]
              , [r].[SalesOrder]
              , [r].[IssMultLotsFlag]
              , [r].[SellingPrice]
              , [MaterialValue] = Coalesce([r].[MaterialValue] , 0) * -1
              , [LabourValue] = Coalesce([r].[LabourValue] , 0)
              , [InputValue] = Coalesce([r].[InputValue] , 0)
              , [SellingValue] = Coalesce([r].[SellingValue] , 0)
              , [Profit] = Coalesce([r].[Profit] , 0)
              , [RemovedValue] = Coalesce([r].[RemovedValue] , 0)
              , [r].[TimeTotal]
              , [r].[MaxSellingPrice]
              , [r].[MinSellingPrice]
        From    [#Results] As [r]
                Left Join [Lookups].[CompanyNames] [CN] On [CN].[Company] =
[r].[DatabaseName]  Collate Latin1_General_BIN;

        Drop Table [#Results];
    End;

GO
EXEC sp_addextendedproperty N'MS_Description', N'high level details of Job, for all
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_Job-Header_AllJobs

```
jobs', 'SCHEMA', N'Report', 'PROCEDURE', N'UspResults_JobHeader_AllJobs', NULL, NULL
GO
```

Uses

[Lookups].[CompanyNames]
[Process].[ExecForEachDB]
[Process].[UspInsert_RedTagLogs]
[Report]

Author: Johnson, Chris

[Report].[UspResults_JobStockDetails]

MS_Description

list of stock related with job

Parameters

Name	Data Type	Max Length (Bytes)
@Company	varchar(max)	max
@RedTagType	char	1
@RedTagUse	varchar(500)	500

SQL Script

```
CREATE Proc [Report].[UspResults_JobStockDetails]
    (
      @Company Varchar(Max)
    , @RedTagType Char(1)
    , @RedTagUse Varchar(500)
    )
As
    Begin
--remove nocount on to speed up query

        Set NoCount On;

        If IsNumeric(@Company) = 0
            Begin
                Select  @Company = Upper(@Company);
            End;

    --Red tag

        Declare @RedTagDB Varchar(255)= Db_Name();
        Exec [Process].[UspInsert_RedTagLogs] @StoredProcDb = 'BlackBox' ,
            @StoredProcSchema = 'Report' ,
            @StoredProcName = 'UspResults_JobStockDetails' ,
            @UsedByType = @RedTagType , @UsedByName = @RedTagUse ,
            @UsedByDb = @RedTagDB;

--list the tables that are to be pulled back from each DB - if they are not found
the script will not be run against that db

        Declare @ListOfTables Varchar(Max) = 'InvMovements,InvMaster';

--create temporary tables to be pulled from different databases, including a column
to id

        Create Table [#InvMovements]
            (
              [DatabaseName] Varchar(150) Collate Latin1_General_BIN
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_Job-StockDetails

```
            , [Job] Varchar(35) Collate Latin1_General_BIN
            , [Warehouse] Varchar(20) Collate Latin1_General_BIN
            , [Bin] Varchar(20) Collate Latin1_General_BIN
            , [StockCode] Varchar(35) Collate Latin1_General_BIN
            , [TrnType] Varchar(5) Collate Latin1_General_BIN
            , [LotSerial] Varchar(50) Collate Latin1_General_BIN
            , [TrnQty] Float
            , [TrnValue] Float
            , [UnitCost] Numeric(20 , 7)
            );
        Create Table [#InvMaster]
            (
              [DatabaseName] Varchar(150) Collate Latin1_General_BIN
            , [StockCode] Varchar(35) Collate Latin1_General_BIN
            , [StockDescription] Varchar(150) Collate Latin1_General_BIN
            , [StockUom] Varchar(10) Collate Latin1_General_BIN
            );


--create script to pull data from each db into the tables

        Declare @SQL1 Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables
            + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'',''))
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
                    Insert  #InvMovements
                                    ( DatabaseName
                                    , Job
                                    , Warehouse
                                    , Bin
                                    , StockCode
                                    , TrnType
                                    , LotSerial
                                    , TrnQty
                                    , TrnValue
                                    ,UnitCost
                                    )
                    Select
                        DatabaseName=@DBCode
                        , Job
                        , Warehouse
                        , Bin
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_Job-StockDetails

```
                        , StockCode
                        , TrnType
                        , LotSerial
                        , TrnQty
                        ,TrnValue
                        ,UnitCost
                    From
                        InvMovements
                    where Job<>'''';
            End
    End';
        Declare @SQL2 Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables
            + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'',''))
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
                Insert #InvMaster
            ( DatabaseName
            , StockCode
            , StockDescription
            ,StockUom
            )
            SELECT DatabaseName=@DBCode
                 , StockCode
                 , Description
                 ,StockUom
            FROM InvMaster
            End
    End';

--execute script against each db, populating the base tables

        Exec [Process].[ExecForEachDB] @cmd = @SQL1;
        Exec [Process].[ExecForEachDB] @cmd = @SQL2;


--Placeholder to create indexes as required


--return results

        Set NoCount Off;
        Select  [IM].[DatabaseName]
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_Job-StockDetails

```
              , [IM].[Job]
              , [IM].[Warehouse]
              , [IM].[Bin]
              , [IM].[StockCode]
              , [IMA].[StockDescription]
              , [OutwardLot] = Case When [IM].[TrnType] = 'R'
                                    Then Substring([IM].[LotSerial] ,
                                                   PatIndex('%[^0]%' ,
                                                            [IM].[LotSerial]) ,
                                                   10)
                                    Else Null
                               End --receipted Lots generated from job

              , [InwardLot] = Case When [IM].[TrnType] <> 'R'
                                        And [IM].[LotSerial] <> ''
                                   Then Substring([IM].[LotSerial] ,
                                                  PatIndex('%[^0]%' ,
                                                           [IM].[LotSerial]) ,
                                                  10)
                                   Else Null
                              End --receipted Lots generated from job

              , [IM].[TrnType]
              , [Quantity] = Sum([IM].[TrnQty] * [TT].[AmountModifier])
              , [Value] = Sum([IM].[TrnValue] * [TT].[AmountModifier])
              , [IMA].[StockUom]
              , [IM].[UnitCost]
        From    [#InvMovements] [IM]
                Left Join [BlackBox].[Lookups].[TrnTypeAmountModifier] [TT]
                    On [TT].[TrnType] = [IM].[TrnType] Collate Latin1_General_BIN
                       And [TT].[Company] = [IM].[DatabaseName] Collate Latin1_-
General_BIN
                Left Join [#InvMaster] [IMA]
                    On [IMA].[StockCode] = [IM].[StockCode] Collate Latin1_General_-
BIN
                       And [IMA].[DatabaseName] = [IM].[DatabaseName] Collate
Latin1_General_BIN
        Group By [IM].[DatabaseName]
              , [IM].[Job]
              , [IM].[Warehouse]
              , [IM].[Bin]
              , [IM].[StockCode]
              , [IMA].[StockDescription]
              , Case When [IM].[TrnType] = 'R'
                     Then Substring([IM].[LotSerial] ,
                                    PatIndex('%[^0]%' , [IM].[LotSerial]) , 10)
                     Else Null
                End
              , Case When [IM].[TrnType] <> 'R'
                          And [IM].[LotSerial] <> ''
                     Then Substring([IM].[LotSerial] ,
                                    PatIndex('%[^0]%' , [IM].[LotSerial]) , 10)
                     Else Null
                End
              , [IM].[TrnType]
              , [IMA].[StockUom]
              , [IM].[UnitCost];
```

Author: Johnson, Chris

```
    End;


GO
EXEC sp_addextendedproperty N'MS_Description', N'list of stock related with job',
'SCHEMA', N'Report', 'PROCEDURE', N'UspResults_JobStockDetails', NULL, NULL
GO
```

Uses

[Lookups].[TrnTypeAmountModifier]
[Process].[ExecForEachDB]
[Process].[UspInsert_RedTagLogs]
[Report]

Author: Johnson, Chris

[Report].[UspResults_JournalDetails]

MS_Description

summary journal info

Parameters

Name	Data Type	Max Length (Bytes)
@Company	varchar(max)	max
@RedTagType	char	1
@RedTagUse	varchar(500)	500

SQL Script

```
CREATE Proc [Report].[UspResults_JournalDetails]
    (
      @Company Varchar(Max)
    , @RedTagType Char(1)
    , @RedTagUse Varchar(500)
    )
As
    Begin
/*
Template designed by Chris Johnson, Prometic Group September 2015
Stored procedure set out to query multiple databases with the same information and
return it in a collated format
*/
        If IsNumeric(@Company) = 0
            Begin
                Select  @Company = Upper(@Company);
            End;

--remove nocount on to speed up query

        Set NoCount On;

--Red tag

        Declare @RedTagDB Varchar(255)= Db_Name();
        Exec [Process].[UspInsert_RedTagLogs] @StoredProcDb = 'BlackBox' ,
            @StoredProcSchema = 'Report' ,
            @StoredProcName = 'UspResults_JournalDetails' ,
            @UsedByType = @RedTagType , @UsedByName = @RedTagUse ,
            @UsedByDb = @RedTagDB;


        Create Table [#Results]
            (
              [Company] Varchar(250)            collate latin1_general_bin
            , [CompanyName] Varchar(250)        collate latin1_general_bin
            , [GlCode] Varchar(35)                collate latin1_general_bin
```

Author: Johnson, Chris

```
            , [GlCodeStart] Varchar(10)            collate latin1_general_bin
            , [GlCodeMid] Varchar(10)             collate latin1_general_bin
            , [GlCodeEnd] Varchar(10)             collate latin1_general_bin
            , [GlCodeDesc] Varchar(250)            collate latin1_general_bin
            , [GlYear] Int
            , [GlPeriod] Int
            , [GlLine] Int
            , [Source] Varchar(250)                collate latin1_general_bin
            , [Journal] Int
            , [JnlDate] Date
            , [Reference] Varchar(250)             collate latin1_general_bin
            , [EntryValue] Numeric(20 , 2)
            , [Comment] Varchar(500)              collate latin1_general_bin
            , [TransactionDate] Date
            , [SubModJournal] Int
            , [SubModInvoiceReg] Int
            , [SubModAssetReg] Int
            , [SubModArInvoice] Varchar(250)     collate latin1_general_bin
            , [SubModApInvoice] Varchar(250)     collate latin1_general_bin
            , [SubModSupplier] Varchar(250)        collate latin1_general_bin
            , [SubModCustomer] Varchar(15)         collate latin1_general_bin
            , [SubModRef] Varchar(30)             collate latin1_general_bin
            , [SubModCheck] Varchar(15)            collate latin1_general_bin
            , [SubModBank] Varchar(15)             collate latin1_general_bin
            , [SubModApBranch] Varchar(10)         collate latin1_general_bin
            , [SubModArBranch] Varchar(10)         collate latin1_general_bin
            , [SubModWh] Varchar(10)              collate latin1_general_bin
            , [SubModStock] Varchar(30)            collate latin1_general_bin
            , [SubModJnlArea] Varchar(10)         collate latin1_general_bin
            , [SubModTransDesc] Varchar(50)        collate latin1_general_bin
            , [SubModAsset] Varchar(30)            collate latin1_general_bin
            , [SubModGrn] Varchar(20)             collate latin1_general_bin
            , [SubModJob] Varchar(20)             collate latin1_general_bin
            , [SubModOperation] Int
            , [SubModWorkCenter] Varchar(20)     collate latin1_general_bin
            , [SubModEmployee] Varchar(20)         collate latin1_general_bin
            , [SubModSalesOrder] Varchar(20)     collate latin1_general_bin
            , [ExtendedComment] Varchar(500)     collate latin1_general_bin
            );

--Placeholder to create indexes as required


--script to combine base data and insert into results table

        Insert  [#Results]
                ( [Company]
                , [CompanyName]
                , [GlCode]
                , [GlCodeStart]
                , [GlCodeMid]
                , [GlCodeEnd]
                , [GlCodeDesc]
                , [GlYear]
                , [GlPeriod]
                , [GlLine]
                , [Source]
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-JournalDetails

```
                , [Journal]
                , [JnlDate]
                , [Reference]
                , [EntryValue]
                , [Comment]
                , [TransactionDate]
                , [SubModJournal]
                , [SubModInvoiceReg]
                , [SubModAssetReg]
                , [SubModArInvoice]
                , [SubModApInvoice]
                , [SubModSupplier]
                , [SubModCustomer]
                , [SubModRef]
                , [SubModCheck]
                , [SubModBank]
                , [SubModApBranch]
                , [SubModArBranch]
                , [SubModWh]
                , [SubModStock]
                , [SubModJnlArea]
                , [SubModTransDesc]
                , [SubModAsset]
                , [SubModGrn]
                , [SubModJob]
                , [SubModOperation]
                , [SubModWorkCenter]
                , [SubModEmployee]
                , [SubModSalesOrder]
                , [ExtendedComment]
                )
                Select  [GT].[Company]
                      , [CN].[CompanyName]
                      , [GT].[GlCode]
                      , [GlCodeStart] = ParseName([GM].[GlCode] , 3)
                      , [GlCodeMid] = ParseName([GM].[GlCode] , 2)
                      , [GlCodeEnd] = ParseName([GM].[GlCode] , 1)
                      , [GlCodeDesc] = LTrim(RTrim([GT].[GlCode])) + ' - '
                        + LTrim(RTrim([GM].[Description]))
                      , [GT].[GlYear]
                      , [GT].[GlPeriod]
                      , [GT].[GlLine]
                      , [Source] = Coalesce([GTS].[SourceDesc] , [GT].[Source])
                      , [GT].[Journal]
                      , [JnlDate] = Cast([GT].[JnlDate] As Date)
                      , [Reference] = Case When [GT].[Reference] = ''
                                           Then Null
                                           Else [GT].[Reference]
                                      End
                      , [GT].[EntryValue]
                      , [Comment] = Case When [GT].[Comment] = '' Then Null
                                         Else [GT].[Comment]
                                    End
                      , [TransactionDate] = Cast([GT].[TransactionDate] As Date)
                      , [SubModJournal] = Case When [GT].[SubModJournal] = 0
                                               Then Null
                                               Else [GT].[SubModJournal]
```

Author: Johnson, Chris

```
                                End
                    , [SubModInvoiceReg] = Case When [GT].[SubModInvoiceReg] = 0
                                               Then Null
                                               Else [GT].[SubModInvoiceReg]
                                           End
                    , [SubModAssetReg] = Case When [GT].[SubModAssetReg] = 0
                                              Then Null
                                              Else [GT].[SubModAssetReg]
                                          End
                    , [SubModArInvoice] = Case When [GT].[SubModArInvoice] = ''
                                               Then Null
                                               Else [GT].[SubModArInvoice]
                                           End
                    , [SubModApInvoice] = Case When [GT].[SubModApInvoice] = ''
                                               Then Null
                                               Else [GT].[SubModApInvoice]
                                           End
                    , [SubModSupplier] = Case When [GT].[SubModSupplier] = ''
                                              Then Null
                                              Else [GT].[SubModSupplier]
                                          End
                    , [SubModCustomer] = Case When [GT].[SubModCustomer] = ''
                                              Then Null
                                              Else [GT].[SubModCustomer]
                                          End
                    , [SubModRef] = Case When [GT].[SubModRef] = ''
                                         Then Null
                                         Else [GT].[SubModRef]
                                     End
                    , [SubModCheck] = Case When [GT].[SubModCheck] = ''
                                           Then Null
                                           Else [GT].[SubModCheck]
                                       End
                    , [SubModBank] = Case When [GT].[SubModBank] = ''
                                          Then Null
                                          Else [GT].[SubModBank]
                                      End
                    , [SubModApBranch] = Case When [GT].[SubModApBranch] = ''
                                              Then Null
                                              Else [GT].[SubModApBranch]
                                          End
                    , [SubModArBranch] = Case When [GT].[SubModArBranch] = ''
                                              Then Null
                                              Else [GT].[SubModArBranch]
                                          End
                    , [SubModWh] = Case When [GT].[SubModWh] = '' Then Null
                                        Else [GT].[SubModWh]
                                    End
                    , [SubModStock] = Case When [GT].[SubModStock] = ''
                                           Then Null
                                           Else [GT].[SubModStock]
                                       End
                    , [SubModJnlArea] = Case When [GT].[SubModJnlArea] = ''
                                             Then Null
                                             Else [GT].[SubModJnlArea]
                                         End
                    , [SubModTransDesc] = Case When [GT].[SubModTransDesc] = ''
```

Author: Johnson, Chris

```
                                                Then Null
                                                Else [GT].[SubModTransDesc]
                                            End
                        , [SubModAsset] = Case When [GT].[SubModAsset] = ''
                                               Then Null
                                               Else [GT].[SubModAsset]
                                          End
                        , [SubModGrn] = Case When [GT].[SubModGrn] = ''
                                             Then Null
                                             Else [GT].[SubModGrn]
                                        End
                        , [SubModJob] = Case When [GT].[SubModJob] = ''
                                             Then Null
                                             Else [GT].[SubModJob]
                                        End
                        , [SubModOperation] = Case When [GT].[SubModOperation] = 0
                                                   Then Null
                                                   Else [GT].[SubModOperation]
                                              End
                        , [SubModWorkCenter] = Case When [GT].[SubModWorkCenter] = ''
                                                    Then Null
                                                    Else [GT].[SubModWorkCenter]
                                               End
                        , [SubModEmployee] = Case When [GT].[SubModEmployee] = ''
                                                  Then Null
                                                  Else [GT].[SubModEmployee]
                                             End
                        , [SubModSalesOrder] = Case When [GT].[SubModSalesOrder] = ''
                                                    Then Null
                                                    Else [GT].[SubModSalesOrder]
                                               End
                        , [ExtendedComment] = Cast([GT].[Journal] As Varchar(10))
                          + ' - ' + Cast([GT].[JnlDate] As Varchar(11))
                          + Case When [GT].[Reference] = '' Then ''
                                 Else ' - ' + [GT].[Reference]
                            End + Case When [GT].[Comment] = '' Then ''
                                       Else ' - ' + [GT].[Comment]
                                  End
                    From    [SysproCompany40].[dbo].[GenTransaction] As [GT] With ( No-
Lock )
                            Inner Join [SysproCompany40].[dbo].[GenMaster] As [GM] On
[GT].[Company] = [GM].[Company]
                                                                   And [GT].[GlCode] =
[GM].[GlCode]
                            Left Join [BlackBox].[Lookups].[CompanyNames] As [CN] On
[CN].[Company] = [GM].[Company]
                            Left Join [BlackBox].[Lookups].[GenTransactionSource]
                            As [GTS] On [GTS].[Source] = [GT].[Source];
--return results

        Select  [Company]
              , [CompanyName]
              , [GlCode]
              , [GlCodeStart]
              , [GlCodeMid]
              , [GlCodeEnd]
              , [GlCodeDesc]
              , [GlYear]
```

Author: Johnson, Chris

```
                , [GlPeriod]
                , [GlLine]
                , [Source]
                , [Journal]
                , [JnlDate]
                , [Reference]
                , [EntryValue]
                , [Comment]
                , [TransactionDate]
                , [SubModJournal]
                , [SubModInvoiceReg]
                , [SubModAssetReg]
                , [SubModArInvoice]
                , [SubModApInvoice]
                , [SubModSupplier]
                , [SubModCustomer]
                , [SubModRef]
                , [SubModCheck]
                , [SubModBank]
                , [SubModApBranch]
                , [SubModArBranch]
                , [SubModWh]
                , [SubModStock]
                , [SubModJnlArea]
                , [SubModTransDesc]
                , [SubModAsset]
                , [SubModGrn]
                , [SubModJob]
                , [SubModOperation]
                , [SubModWorkCenter]
                , [SubModEmployee]
                , [SubModSalesOrder]
                , [ExtendedComment]
        From    [#Results]
        Where   Case When @Company = 'ALL' Then 'ALL' Else [Company] End = @Company;

    End;

GO
EXEC sp_addextendedproperty N'MS_Description', N'summary journal info', 'SCHEMA',
N'Report', 'PROCEDURE', N'UspResults_JournalDetails', NULL, NULL
GO
```

Uses

[Lookups].[CompanyNames]
[Lookups].[GenTransactionSource]
[Process].[UspInsert_RedTagLogs]
[Report]

Author: Johnson, Chris

[Report].[UspResults_Labour]

MS_Description

not in use

Parameters

Name	Data Type	Max Length (Bytes)
@Company	varchar(max)	max
@RedTagType	char	1
@RedTagUse	varchar(500)	500

SQL Script

```
CREATE Proc [Report].[UspResults_Labour]
    (
      @Company Varchar(Max)
    , @RedTagType Char(1)
    , @RedTagUse Varchar(500)
    )
As
    Begin
/*
Template designed by Chris Johnson, Prometic Group September 2015
Stored procedure set out to query multiple databases with the same information and
return it in a collated format        ///
--Exec [Report].[UspResults_Labour] 10
*/
        If IsNumeric(@Company) = 0
            Begin
                Select  @Company = Upper(@Company);
            End;

--remove nocount on to speed up query

        Set NoCount On;
--Red tag

        Declare @RedTagDB Varchar(255)= Db_Name();
        Exec [Process].[UspInsert_RedTagLogs] @StoredProcDb = 'BlackBox' ,
            @StoredProcSchema = 'Report' ,
            @StoredProcName = 'UspResults_Labour' , @UsedByType = @RedTagType ,
            @UsedByName = @RedTagUse , @UsedByDb = @RedTagDB;
--list the tables that are to be pulled back from each DB - if they are not found
the script will not be run against that db

        Declare @ListOfTables Varchar(Max) = 'AssetDepreciation,TblApTerms';

--create temporary tables to be pulled from different databases, including a column
to id
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-Labour

```
        Create Table [#WipLabJnl]
            (
              [DatabaseName] Varchar(150)    Collate Latin1_General_BIN
            , [EntryMonth] Date
            , [EntryDate] DateTime2
            , [Employee] Varchar(20)        collate latin1_general_bin
            , [RunTime] Numeric(20 , 7)
            , [RunTimeRate] Numeric(20 , 7)
            , [SetUpTime] Numeric(20 , 7)
            , [SetUpRate] Numeric(20 , 7)
            , [StartUpTime] Numeric(20 , 7)
            , [StartUpRate] Numeric(20 , 7)
            , [TeardownTime] Numeric(20 , 7)
            , [TeardownRate] Numeric(20 , 7)
            , [LabourValue] Numeric(20 , 7)
            , [FixedOhRate] Numeric(20 , 7)
            , [VariableOhRate] Numeric(20 , 7)
            , [WorkCentre] Varchar(150)        collate latin1_general_bin
            , [Job] Varchar(50)                collate latin1_general_bin
            );

--create script to pull data from each db into the tables

        Declare @SQL Varchar(Max) = 'USE [?];
Declare @DB varchar(150),@DBCode varchar(150)
Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end
IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
BEGIN'
            + --only companies selected in main run, or if companies selected then
all

            '
    IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
        Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables + '''
                , @RequiredCountOfTables INT
                , @ActualCountOfTables INT'
            + --count number of tables requested (number of commas plus one)

            '
        Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')'
            + --Count of the tables requested how many exist in the db

            '
        Select @ActualCountOfTables = COUNT(1) FROM sys.tables
        Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'','')) '
            + --only if the count matches (all the tables exist in the requested db)
then run the script

            '
        If @ActualCountOfTables=@RequiredCountOfTables
        BEGIN
                Insert [#WipLabJnl]
        ( [DatabaseName], [EntryMonth], [EntryDate]
        , [Employee], [RunTime], [RunTimeRate]
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-Labour

```
        , [SetUpTime], [SetUpRate]
        , [StartUpTime], [StartUpRate]
        , [TeardownTime], [TeardownRate]
        , [LabourValue], [FixedOhRate], [VariableOhRate]
        ,WorkCentre,[Job]
        )
SELECT  @DBCode,[EntryMonth] = DATEADD(Day,-(DAY([wlj].[EntryDate])-1),[wlj].[Entry-
Date]),[EntryDate]
,[Employee],[RunTime],[RunTimeRate]
,[SetUpTime],[SetUpRate]
,[StartUpTime],[StartUpRate]
,[TeardownTime],[TeardownRate]
,[LabourValue],[FixedOhRate],[VariableOhRate]
,WorkCentre,[Job]
FROM [WipLabJnl] As [wlj] Order By EntryMonth Desc
        End
End';

--Enable this function to check script changes (try to run script directly against
db manually)


--execute script against each db, populating the base tables

        Exec [Process].[ExecForEachDB] @cmd = @SQL;

--define the results you want to return


--Placeholder to create indexes as required


--script to combine base data and insert into results table


--return results

        Select  [wlj].[DatabaseName]
              , [cn].[CompanyName]
              , [wlj].[EntryMonth]
              , [wlj].[EntryDate]
              , [wlj].[Employee]
              , [wlj].[RunTime]
              , [wlj].[RunTimeRate]
              , [wlj].[SetUpTime]
              , [wlj].[SetUpRate]
              , [wlj].[StartUpTime]
              , [wlj].[StartUpRate]
              , [wlj].[TeardownTime]
              , [wlj].[TeardownRate]
              , [wlj].[LabourValue]
              , [wlj].[FixedOhRate]
              , [wlj].[VariableOhRate]
              , [wlj].[WorkCentre]
              , [wlj].[Job]
        From    [#WipLabJnl] As [wlj]
                Left Join [Lookups].[CompanyNames] As [cn] On [wlj].[DatabaseName] =
[cn].[Company];
```

Author: Johnson, Chris

```
    End;


GO
EXEC sp_addextendedproperty N'MS_Description', N'not in use', 'SCHEMA', N'Report',
'PROCEDURE', N'UspResults_Labour', NULL, NULL
GO
```

Uses

[Lookups].[CompanyNames]
[Process].[ExecForEachDB]
[Process].[UspInsert_RedTagLogs]
[Report]

[Report].[UspResults_LabourDetails]

MS_Description

list of labour

Parameters

Name	Data Type	Max Length (Bytes)
@Company	varchar(max)	max
@RedTagType	char	1
@RedTagUse	varchar(500)	500

SQL Script

```
CREATE Proc [Report].[UspResults_LabourDetails]
    (
      @Company Varchar(Max)
    , @RedTagType Char(1)
    , @RedTagUse Varchar(500)
    )
As
    Begin
/*
Template designed by Chris Johnson, Prometic Group September 2015
Stored procedure set out to query multiple databases with the same information and
return it in a collated format
--exec Report.UspResults_LabourDetails 10
*/
        If IsNumeric(@Company) = 0
            Begin
                Select  @Company = Upper(@Company);
            End;


--remove nocount on to speed up query

        Set NoCount On;

--Red tag

        Declare @RedTagDB Varchar(255)= Db_Name();
        Exec [Process].[UspInsert_RedTagLogs] @StoredProcDb = 'BlackBox' ,
            @StoredProcSchema = 'Report' ,
            @StoredProcName = 'UspResults_LabourDetails' ,
            @UsedByType = @RedTagType , @UsedByName = @RedTagUse ,
            @UsedByDb = @RedTagDB;

--list the tables that are to be pulled back from each DB - if they are not found
the script will not be run against that db

        Declare @ListOfTables Varchar(Max) = 'WipLabJnl';
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-LabourDetails

```
--create temporary tables to be pulled from different databases, including a column
to id

        Create Table [#WipLabJnl]
            (
              [DatabaseName] Varchar(150) Collate Latin1_General_BIN
            , [Job] Varchar(35) Collate Latin1_General_BIN
            , [TrnYear] Int
            , [TrnMonth] Int
            , [Machine] Varchar(150) Collate Latin1_General_BIN
            , [WorkCentre] Varchar(35) Collate Latin1_General_BIN
            , [Employee] Varchar(35) Collate Latin1_General_BIN
            , [SetUpRate] Numeric(20 , 7)
            , [RunTimeRate] Numeric(20 , 7)
            , [FixedOhRate] Numeric(20 , 7)
            , [VariableOhRate] Numeric(20 , 7)
            , [StartUpRate] Numeric(20 , 7)
            , [TeardownRate] Numeric(20 , 7)
            , [LabourValue] Numeric(20 , 7)
            , [EntryDate] DateTime2
            , [RunTime] Numeric(20 , 7)
            , [SetUpTime] Numeric(20 , 7)
            , [StartUpTime] Numeric(20 , 7)
            , [TeardownTime] Numeric(20 , 7)
            );



--create script to pull data from each db into the tables

        Declare @SQL Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end'
            + --Only query DBs beginning SysProCompany

            '
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN'
            + --only companies selected in main run, or if companies selected then
all

            '
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT'
            + --count number of tables requested (number of commas plus one)

            '
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')'
            + --Count of the tables requested how many exist in the db

            '
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-LabourDetails

```
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'','')) '
            + --only if the count matches (all the tables exist in the requested db)
then run the script

            '
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
                    Insert #WipLabJnl
                                ( DatabaseName, TrnYear
                                , TrnMonth, Machine
                                , WorkCentre, Employee
                                , SetUpRate, RunTimeRate
                                , FixedOhRate, VariableOhRate
                                , StartUpRate, TeardownRate
                                , LabourValue, EntryDate
                                , Job
                                , RunTime, SetUpTime
                                , StartUpTime, TeardownTime
                                )
                        SELECT DatabaseName = @DBCode
                             , TrnYear
                             , TrnMonth, Machine
                             , WorkCentre, Employee
                             , SetUpRate, RunTimeRate
                             , FixedOhRate, VariableOhRate
                             , StartUpRate, TeardownRate
                             , LabourValue, EntryDate
                             , Job
                             , RunTime, SetUpTime
                             , StartUpTime, TeardownTime
                    FROM WipLabJnl
            End
    End';

--Enable this function to check script changes (try to run script directly against
db manually)

--Print @SQL


--execute script against each db, populating the base tables

        Exec [Process].[ExecForEachDB] @cmd = @SQL;

--define the results you want to return

        Create Table [#LabourDetailsResults]
            (
              [Company] Varchar(150) Collate Latin1_General_BIN
            , [Job] Varchar(35) Collate Latin1_General_BIN
            , [TrnYear] Int
            , [TrnMonth] Int
            , [Machine] Varchar(150) Collate Latin1_General_BIN
            , [WorkCentre] Varchar(35) Collate Latin1_General_BIN
            , [Employee] Varchar(35) Collate Latin1_General_BIN
            , [SetUpRate] Numeric(20 , 7)
```

Author: Johnson, Chris

```
            , [RunTimeRate] Numeric(20 , 7)
            , [FixedOhRate] Numeric(20 , 7)
            , [VariableOhRate] Numeric(20 , 7)
            , [StartUpRate] Numeric(20 , 7)
            , [TeardownRate] Numeric(20 , 7)
            , [LabourValue] Numeric(20 , 7)
            , [EntryDate] DateTime2
            , [RunTime] Numeric(20 , 7)
            , [SetUpTime] Numeric(20 , 7)
            , [StartUpTime] Numeric(20 , 7)
            , [TeardownTime] Numeric(20 , 7)
            );

--Placeholder to create indexes as required

--create NonClustered Index Index_Name On #Table1 (DatabaseName) Include (Column-
Name)


--script to combine base data and insert into results table

        Insert  [#LabourDetailsResults]
                ( [Company]
                , [Job]
                , [TrnYear]
                , [TrnMonth]
                , [Machine]
                , [WorkCentre]
                , [Employee]
                , [SetUpRate]
                , [RunTimeRate]
                , [FixedOhRate]
                , [VariableOhRate]
                , [StartUpRate]
                , [TeardownRate]
                , [LabourValue]
                , [EntryDate]
                , [RunTime]
                , [SetUpTime]
                , [StartUpTime]
                , [TeardownTime]
                )
                Select  [Company] = [cn].[CompanyName]
                      , [WJL].[Job]
                      , [WJL].[TrnYear]
                      , [WJL].[TrnMonth]
                      , [WJL].[Machine]
                      , [WJL].[WorkCentre]
                      , [WJL].[Employee]
                      , [WJL].[SetUpRate]
                      , [WJL].[RunTimeRate]
                      , [WJL].[FixedOhRate]
                      , [WJL].[VariableOhRate]
                      , [WJL].[StartUpRate]
                      , [WJL].[TeardownRate]
                      , [WJL].[LabourValue]
                      , [WJL].[EntryDate]
```

Author: Johnson, Chris

```
                      , [WJL].[RunTime]
                      , [WJL].[SetUpTime]
                      , [WJL].[StartUpTime]
                      , [WJL].[TeardownTime]
                From    [#WipLabJnl] [WJL]
                        Left Join [Lookups].[CompanyNames] As [cn]
                            On [WJL].[DatabaseName] = [cn].[Company];

--return results

        Select  [r].[Company]
              , [r].[Job]
              , [r].[TrnYear]
              , [r].[TrnMonth]
              , [r].[Machine]
              , [r].[WorkCentre]
              , [r].[Employee]
              , [r].[SetUpRate]
              , [r].[RunTimeRate]
              , [r].[FixedOhRate]
              , [r].[VariableOhRate]
              , [r].[StartUpRate]
              , [r].[TeardownRate]
              , [r].[LabourValue]
              , [r].[RunTime]
              , [r].[SetUpTime]
              , [r].[StartUpTime]
              , [r].[TeardownTime]
        From    [#LabourDetailsResults] As [r];

        Drop Table [#LabourDetailsResults];
    End;

GO
EXEC sp_addextendedproperty N'MS_Description', N'list of labour', 'SCHEMA',
N'Report', 'PROCEDURE', N'UspResults_LabourDetails', NULL, NULL
GO
```

Uses

[Lookups].[CompanyNames]
[Process].[ExecForEachDB]
[Process].[UspInsert_RedTagLogs]
[Report]

Author: Johnson, Chris

[Report].[UspResults_LabourGlCodes]

MS_Description

not in use

Parameters

Name	Data Type	Max Length (Bytes)
@Company	varchar(max)	max
@RedTagType	char	1
@RedTagUse	varchar(500)	500

SQL Script

```
CREATE Proc [Report].[UspResults_LabourGlCodes]
    (
      @Company Varchar(Max)
    , @RedTagType Char(1)
    , @RedTagUse Varchar(500)
    )
As
    Begin
/*
Template designed by Chris Johnson, Prometic Group September 2015
Stored procedure set out to query multiple databases with the same information and
return it in a collated format
--Exec [Report].[UspResults_LabourGlCodes]  @Company ='10'
*/
        If IsNumeric(@Company) = 0
            Begin
                Select  @Company = Upper(@Company);
            End;

--remove nocount on to speed up query

        Set NoCount On;
--Red tag

        Declare @RedTagDB Varchar(255)= Db_Name();
        Exec [Process].[UspInsert_RedTagLogs] @StoredProcDb = 'BlackBox' ,
            @StoredProcSchema = 'Report' ,
            @StoredProcName = 'UspResults_LabourGlCodes' ,
            @UsedByType = @RedTagType , @UsedByName = @RedTagUse ,
            @UsedByDb = @RedTagDB;
--list the tables that are to be pulled back from each DB - if they are not found
the script will not be run against that db

        Declare @ListOfTables Varchar(Max) = 'AssetDepreciation,TblApTerms';

        Select  [GlMonth] = DateAdd(Month , [gjd].[GlPeriod] - 1 ,
                                    DateAdd(Year , [gjd].[GlYear] - 2000 ,
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-LabourGlCodes

```
                                        Cast('2000-01-01' As Date)))
              , [gjd].[GlYear]
              , [gjd].[GlPeriod]
              , [Job] = [gjd].[SubModJob]
              , [EntryDate] = Cast([gjd].[EntryDate] As Date)
              , [gjd].[Journal]
              , [gjd].[GlCode]
              , [gm2].[Description]
              , [Comment] = Case When [gjd].[Comment] = '' Then Null
                                 Else [gjd].[Comment]
                            End
              , [JournalValue] = [gjd].[EntryValue]
                * Case When [gjd].[EntryType] = 'D' Then 1
                       When [gjd].[EntryType] = 'C' Then -1
                       Else 0
                  End
              , [gm].[Mapping1]
              , [gm].[Mapping2]
              , [gm].[Mapping3]
              , [gm].[Mapping4]
              , [gm].[Mapping5]
              , [FixedOhRate] = Cast(Null As Numeric(20 , 7))
              , [VariableOhRate] = Cast(Null As Numeric(20 , 7))
              , [WorkCentre] = Cast(Null As Varchar(150))
              , [Employee] = Cast(Null As Varchar(150))
              , [RunTime] = Cast(Null As Numeric(20 , 7))
              , [RunTimeRate] = Cast(Null As Numeric(20 , 7))
              , [SetUpTime] = Cast(Null As Numeric(20 , 7))
              , [SetUpRate] = Cast(Null As Numeric(20 , 7))
              , [StartUpTime] = Cast(Null As Numeric(20 , 7))
              , [StartUpRate] = Cast(Null As Numeric(20 , 7))
              , [TeardownTime] = Cast(Null As Numeric(20 , 7))
              , [TeardownRate] = Cast(Null As Numeric(20 , 7))
              , [LabourValue] = Cast(Null As Numeric(20 , 7))
        From    [SysproCompany10]..[GenJournalDetail] As [gjd]
                Left Join [Lookups].[GLMapping] As [gm] On [gm].[GlCode] = [gjd].[Gl-
Code]
                                                           And [gm].[Company] = '10'
                Left Join [SysproCompany40]..[GenMaster] As [gm2] On [gm2].[GlCode]
= [gjd].[GlCode]
                                                             And [gm2].[Company] =
'10'
        Union All
        Select  [GlMonth] = DateAdd(Month , [wlj].[GlPeriod] - 1 ,
                                    DateAdd(Year , [wlj].[GlYear] - 2000 ,
                                            Cast('2000-01-01' As Date)))
              , [wlj].[GlYear]
              , [wlj].[GlPeriod]
              , [wlj].[Job]
              , [EntryDate] = Cast([wlj].[EntryDate] As Date)
              , [wlj].[Journal]
              , [GlCode] = 'Labour'
              , [Description] = Null
              , [Comment] = Null
              , [JournalValue] = Null
              , [Mapping1] = 'Labour'
              , [Mapping2] = 'Labour'
              , [Mapping3] = 'Labour'
```

Author: Johnson, Chris

```
                , [Mapping4] = 'Labour'
                , [Mapping5] = 'Labour'
                , [wlj].[FixedOhRate]
                , [wlj].[VariableOhRate]
                , [wlj].[WorkCentre]
                , [wlj].[Employee]
                , [RunTime] = Sum([wlj].[RunTime])
                , [wlj].[RunTimeRate]
                , [SetUpTime] = Sum([wlj].[SetUpTime])
                , [wlj].[SetUpRate]
                , [StartUpTime] = Sum([wlj].[StartUpTime])
                , [wlj].[StartUpRate]
                , [TeardownTime] = Sum([wlj].[TeardownTime])
                , [wlj].[TeardownRate]
                , [LabourValue] = Sum([wlj].[LabourValue])
        From    [SysproCompany10]..[WipLabJnl] As [wlj]
        Group By DateAdd(Month , [wlj].[GlPeriod] - 1 ,
                         DateAdd(Year , [wlj].[GlYear] - 2000 ,
                                 Cast('2000-01-01' As Date)))
                , Cast([EntryDate] As Date)
                , [wlj].[GlPeriod]
                , [wlj].[GlYear]
                , [wlj].[Journal]
                , [wlj].[Employee]
                , [wlj].[RunTimeRate]
                , [wlj].[SetUpRate]
                , [wlj].[StartUpRate]
                , [wlj].[TeardownRate]
                , [wlj].[FixedOhRate]
                , [wlj].[VariableOhRate]
                , [wlj].[WorkCentre]
                , [wlj].[Job]
        Order By [GlMonth] Desc;

    End;

GO
EXEC sp_addextendedproperty N'MS_Description', N'not in use', 'SCHEMA', N'Report',
'PROCEDURE', N'UspResults_LabourGlCodes', NULL, NULL
GO
```

Uses

[Lookups].[GLMapping]
[Process].[UspInsert_RedTagLogs]
[Report]

Author: Johnson, Chris

[Report].[UspResults_LabourLoggedPerISOWeek]

MS_Description

labour logged split by ISO week

Parameters

Name	Data Type	Max Length (Bytes)
@Company	varchar(max)	max
@Year	int	4
@MaxDate	date	3
@RedTagType	char	1
@RedTagUse	varchar(500)	500

SQL Script

```
CREATE Proc [Report].[UspResults_LabourLoggedPerISOWeek]
    (
      @Company Varchar(Max)
    , @Year Int
    , @MaxDate Date
    , @RedTagType Char(1)
    , @RedTagUse Varchar(500)
    )
As
    Begin
/*
Template designed by Chris Johnson, Prometic Group September 2015
Stored procedure set out to query multiple databases with the same information and
return it in a collated format
*/
        Set NoCount On;

        If IsNumeric(@Company) = 0
            Begin
                Select  @Company = Upper(@Company);
            End;

--Red tag

        Declare @RedTagDB Varchar(255)= Db_Name();
        Exec [Process].[UspInsert_RedTagLogs] @StoredProcDb = 'BlackBox' ,
            @StoredProcSchema = 'Report' ,
            @StoredProcName = 'UspResults_LabourLoggedPerISOWeek' ,
            @UsedByType = @RedTagType , @UsedByName = @RedTagUse ,
            @UsedByDb = @RedTagDB;

--list the tables that are to be pulled back from each DB - if they are not found
the script will not be run against that db
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-LabourLoggedPerISOWeek

```
        Declare @ListOfTables Varchar(Max) = 'WipLabJnl';

--create temporary tables to be pulled from different databases, including a column to id

        Create Table [#WipLabJnl]
            (
              [DatabaseName] Varchar(150)        collate latin1_general_bin
            , [Employee] Varchar(20)            collate latin1_general_bin
            , [TrnYear] Int
            , [EntryDate] Date
            , [RunTime] Numeric(20 , 6)
            , [SetUpTime] Numeric(20 , 6)
            , [StartUpTime] Numeric(20 , 6)
            , [TeardownTime] Numeric(20 , 6)
            );

        Create Index [WLJ_DB_Emp_TY_ED] On [#WipLabJnl] ([Database-Name],[Employee],[TrnYear],[EntryDate]);

--create script to pull data from each db into the tables

        Declare @SQL Varchar(Max) = 'USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-Name(),len(db_Name())-13) else null end
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            BEGIN
                Insert [#WipLabJnl]
                        ( [DatabaseName]
                        , [Employee]
                        , [TrnYear]
                        , [EntryDate]
                        , [RunTime]
                        , [SetUpTime]
                        , [StartUpTime]
                        , [TeardownTime]
                        )
                SELECT [DatabaseName]=@DBCode
                     , [WLJ].[Employee]
                     , [WLJ].[TrnYear]
                     , [WLJ].[EntryDate]
                     , [WLJ].[RunTime]
                     , [WLJ].[SetUpTime]
                     , [WLJ].[StartUpTime]
                     , [WLJ].[TeardownTime]
                FROM [WipLabJnl] [WLJ]
            End
    End';

--Enable this function to check script changes (try to run script directly against db manually)

--Print @SQL
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-LabourLoggedPerISOWeek

```
--execute script against each db, populating the base tables

        Exec [Process].[ExecForEachDB_WithTableCheck] @cmd = @SQL ,
            @SchemaTablesToCheck = @ListOfTables;


--define the results you want to return

        Create Table [#Results]
            (
              [DatabaseName] Varchar(150)        collate latin1_general_bin
            , [Employee] Varchar(20)            collate latin1_general_bin
            , [TrnYear] Int
            , [ISOWeek] Int
            , [RunTime] Numeric(20 , 6)
            , [SetUpTime] Numeric(20 , 6)
            , [StartUpTime] Numeric(20 , 6)
            , [TeardownTime] Numeric(20 , 6)
            , [TestForFullHoursAccounted] As Case When [RunTime] > 2250
                                                  Then 'More'
                                                  When [RunTime] < 2250
                                                  Then 'Less'
                                                  Else 'Correct'
                                             End
            );

--Placeholder to create indexes as required


--script to combine base data and insert into results table

        Insert  [#Results]
                ( [DatabaseName]
                , [Employee]
                , [TrnYear]
                , [ISOWeek]
                , [RunTime]
                , [SetUpTime]
                , [StartUpTime]
                , [TeardownTime]
                )
                Select  [wlj].[DatabaseName]
                      , [wlj].[Employee]
                      , [wlj].[TrnYear]
                      , [ISOWeek] = DatePart(iso_week , [wlj].[EntryDate])
                      , [RunTime] = Sum([wlj].[RunTime])
                      , [SetUpTime] = Sum([wlj].[SetUpTime])
                      , [StartUpTime] = Sum([wlj].[StartUpTime])
                      , [TeardownTime] = Sum([wlj].[TeardownTime])
                From    [#WipLabJnl] As [wlj]
                Where   [wlj].[TrnYear] = @Year
                        And [wlj].[EntryDate] <= @MaxDate
                Group By [wlj].[DatabaseName]
                      , DatePart(iso_week , [wlj].[EntryDate])
                      , [wlj].[Employee]
                      , [wlj].[TrnYear];
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-LabourLoggedPerISOWeek

```
        Set NoCount Off;
--return results

        Select  [CN].[CompanyName]
              , [CN].[ShortName]
              , [CN].[Currency]
              , [r].[Employee]
              , [r].[TrnYear]
              , [r].[ISOWeek]
              , [r].[RunTime]
              , [r].[SetUpTime]
              , [r].[StartUpTime]
              , [r].[TeardownTime]
              , [r].[TestForFullHoursAccounted]
        From    [#Results] [r]
                Left Join [Lookups].[CompanyNames] [CN]
                    On [CN].[Company] = [r].[DatabaseName];

    End;

GO
EXEC sp_addextendedproperty N'MS_Description', N'labour logged split by ISO week',
'SCHEMA', N'Report', 'PROCEDURE', N'UspResults_LabourLoggedPerISOWeek', NULL, NULL
GO
```

Uses

[Lookups].[CompanyNames]
[Process].[ExecForEachDB_WithTableCheck]
[Process].[UspInsert_RedTagLogs]
[Report]

Author: Johnson, Chris

[Report].[UspResults_ListAllCompanies]

MS_Description

list of all companies available

SQL Script

```
CREATE Proc [Report].[UspResults_ListAllCompanies]
As
    Begin
/*
Template designed by Chris Johnson, Prometic Group September 2015
Stored procedure created by Chris Johnson, Prometic Group September 2015 to return a list of all SysPro databases
for use in report dropdowns
*/
        Declare @Company Varchar(3)= 'All';
--remove nocount on to speed up query

        Set NoCount On;

--list the tables that are to be pulled back from each DB - if they are not found the script will not be run against that db

        Declare @ListOfTables Varchar(Max) = '';
        --no tables required as querying variables


--create temporary tables to be pulled from different databases, including a column to id

        Create Table [#Table1]
            (
              [CompanyName] Varchar(150)     collate Latin1_General_BIN
            );

--create script to pull data from each db into the tables

        Declare @SQL Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-Name(),len(db_Name())-13) else null end'
            + --Only query DBs beginning SysProCompany

            '
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN'
            + --only companies selected in main run, or if companies selected then all

            '
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_List-AllCompanies

```
            --Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables + '''
            --        , @RequiredCountOfTables INT
            --        , @ActualCountOfTables INT'
            + ----removed table count from template

            '--
            --Select @RequiredCountOfTables=BlackBox.dbo.[Udf_CountText]('','',@List-
OfTables)+1'
            + ----removed table count from template

            '
            --Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            --Where name In (Select Value Collate Latin1_General_BIN From Local-
Dev.dbo.udfSplitString(@ListOfTables,'','')) '
            + ----removed table count from template

            '
            --If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
                Insert #Table1
                    ( CompanyName )
                Select @DBCode

            End
    End';

--Enable this function to check script changes (try to run script directly against
db manually)

        Print @SQL;

--execute script against each db, populating the base tables

        Exec [Process].[ExecForEachDB] @cmd = @SQL;

--define the results you want to return

        Create Table [#Results]
            (
              [CompanyName] Varchar(150)
            );

--Placeholder to create indexes as required


--script to combine base data and insert into results table

        Insert  [#Results]
                ( [CompanyName]
                )
                Select  [CompanyName]
                From    [#Table1]
                Union
                Select  'All';

--return results

        Select  [CompanyName]
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_List-AllCompanies

```
        From    [#Results]
        Order By Case When [CompanyName] = 'All' Then 0
                      When IsNumeric([CompanyName]) = 1 Then 1
                      Else 2
                 End Asc
               , [CompanyName] Asc;


    End;

GO
EXEC sp_addextendedproperty N'MS_Description', N'list of all companies available',
'SCHEMA', N'Report', 'PROCEDURE', N'UspResults_ListAllCompanies', NULL, NULL
GO
```

Uses

[Process].[ExecForEachDB]
[Report]

Author: Johnson, Chris

[Report].[UspResults_LotsRetesting]

MS_Description

list of lots and when they need to be retested

Parameters

Name	Data Type	Max Length (Bytes)
@Company	varchar(max)	max
@RedTagType	char	1
@RedTagUse	varchar(500)	500

SQL Script

```
CREATE Proc [Report].[UspResults_LotsRetesting]
    (
      @Company Varchar(Max)
    , @RedTagType Char(1)
    , @RedTagUse Varchar(500)
    )
As
    Begin
/*
Template designed by Chris Johnson, Prometic Group September 2015
Stored procedure set out to query multiple databases with the same information and
return it in a collated format
--Exec  [Report].[UspResults_LotsRetesting] 10
*/
        If IsNumeric(@Company) = 0
            Begin
                Select  @Company = Upper(@Company);
            End;


--remove nocount on to speed up query

        Set NoCount On;

--Red tag

        Declare @RedTagDB Varchar(255)= Db_Name();
        Exec [Process].[UspInsert_RedTagLogs] @StoredProcDb = 'BlackBox' ,
            @StoredProcSchema = 'Report' ,
            @StoredProcName = 'UspResults_LotsRetesting' ,
            @UsedByType = @RedTagType , @UsedByName = @RedTagUse ,
            @UsedByDb = @RedTagDB;

--list the tables that are to be pulled back from each DB - if they are not found
the script will not be run against that db

        Declare @ListOfTables Varchar(Max) = 'InvMaster,InvWhControl,LotDetail';
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_Lots-Retesting

```
--create temporary tables to be pulled from different databases, including a column
to id

        Create Table [#Lots]
            (
              [DatabaseName] Varchar(150) Collate Latin1_General_BIN
            , [JobPurchOrder] Varchar(50) Collate Latin1_General_BIN
            , [Lot] Varchar(50) Collate Latin1_General_BIN
            , [StockCode] Varchar(50) Collate Latin1_General_BIN
            );
        Create Table [#InvMaster]
            (
              [DatabaseName] Varchar(150) Collate Latin1_General_BIN
            , [StockCode] Varchar(35) Collate Latin1_General_BIN
            , [Description] Varchar(150) Collate Latin1_General_BIN
            );
        Create Table [#InvWhControl]
            (
              [DatabaseName] Varchar(150) Collate Latin1_General_BIN
            , [Warehouse] Varchar(10) Collate Latin1_General_BIN
            , [Description] Varchar(150) Collate Latin1_General_BIN
            );
        Create Table [#LotDetail]
            (
              [DatabaseName] Varchar(150) Collate Latin1_General_BIN
            , [StockCode] Varchar(35) Collate Latin1_General_BIN
            , [Warehouse] Varchar(10) Collate Latin1_General_BIN
            , [Lot] Varchar(35) Collate Latin1_General_BIN
            , [Bin] Varchar(10) Collate Latin1_General_BIN
            , [QtyOnHand] Decimal(20 , 7)
            , [ExpiryDate] DateTime2
            );
        Create Table [#LatestLots]
            (
              [DatabaseName] Varchar(150) Collate Latin1_General_BIN
            , [StockCode] Varchar(30)  Collate Latin1_General_BIN
            , [Lot] Varchar(35) Collate Latin1_General_BIN
            , [UnitCost] Numeric(20 , 8)
            );

--create script to pull data from each db into the tables

        Declare @SQL1 Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end'
            + --Only query DBs beginning SysProCompany

            '
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN'
            + --only companies selected in main run, or if companies selected then
all

            '
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_Lots-Retesting

```
            + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT'
            + --count number of tables requested (number of commas plus one)

            '
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')'
            + --Count of the tables requested how many exist in the db

            '
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'','')) '
            + --only if the count matches (all the tables exist in the requested db)
then run the script

            '
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
             Insert  [#Lots]
                ( [DatabaseName]
                , [JobPurchOrder]
                , [Lot]
                , [StockCode]
                )
              Select Distinct
                    [DatabaseName] = @DBCode
                  , JobPurchOrder
                  , Lot
                  , LT.[StockCode]
                From
                    dbo.LotTransactions LT
                Where
                    TrnType = ''R''
                    And JobPurchOrder <> ''''
                    and Warehouse <>''RM''
            End
    End';
        Declare @SQL2 Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end'
            + --Only query DBs beginning SysProCompany

            '
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN'
            + --only companies selected in main run, or if companies selected then
all

            '
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables + '''
                    , @RequiredCountOfTables INT
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_Lots-Retesting

```
                    , @ActualCountOfTables INT'
            + --count number of tables requested (number of commas plus one)

            '
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')'
            + --Count of the tables requested how many exist in the db

            '
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'','')) '
            + --only if the count matches (all the tables exist in the requested db)
then run the script

            '
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
             Insert  [#InvMaster]
                ( [DatabaseName]
                , [StockCode]
                , [Description]
                )
             Select Distinct [DatabaseName] = @DBCode
                  , [im].[StockCode]
                  , [im].[Description]
              From [InvMaster] As [im]
            End
    End';
        Declare @SQL3 Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end'
            + --Only query DBs beginning SysProCompany

            '
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN'
            + --only companies selected in main run, or if companies selected then
all

            '
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT'
            + --count number of tables requested (number of commas plus one)

            '
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')'
            + --Count of the tables requested how many exist in the db

            '
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'','')) '
```

Author: Johnson, Chris

```
            + --only if the count matches (all the tables exist in the requested db)
then run the script

            '
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
                Insert [#InvWhControl]
                    ( [DatabaseName]
                    , [Warehouse]
                    , [Description] )
                SELECT [DatabaseName]=@DBCode
                    , [iwc].[Warehouse]
                    , [iwc].[Description]
                FROM [InvWhControl] As [iwc]
            End
    End';
        Declare @SQL4 Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end'
            + --Only query DBs beginning SysProCompany

            '
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN'
            + --only companies selected in main run, or if companies selected then
all

            '
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT'
            + --count number of tables requested (number of commas plus one)

            '
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')'
            + --Count of the tables requested how many exist in the db

            '
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'','')) '
            + --only if the count matches (all the tables exist in the requested db)
then run the script

            '
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
                Insert [#LotDetail]
                        ( [DatabaseName]
                        , [StockCode]
                        , [Warehouse]
                        , [Lot]
                        , [Bin]
```

Author: Johnson, Chris

```
                        , [QtyOnHand]
                        , [ExpiryDate]
                        )
                SELECT [DatabaseName]=@DBCode
                     , [ld].[StockCode]
                     , [ld].[Warehouse]
                     , [ld].[Lot]
                     , [ld].[Bin]
                     , [ld].[QtyOnHand]
                     , [ld].[ExpiryDate] FROM [LotDetail] As [ld]
            End
    End';
        Declare @SQLLatestLots Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end'
            + --Only query DBs beginning SysProCompany

            '
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN'
            + --only companies selected in main run, or if companies selected then
all

            '
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT'
            + --count number of tables requested (number of commas plus one)

            '
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')'
            + --Count of the tables requested how many exist in the db

            '
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'','')) '
            + --only if the count matches (all the tables exist in the requested db)
then run the script

            '
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
                        Insert [#LatestLots]
                ( [DatabaseName]
                , [StockCode]
                , [Lot]
                , [UnitCost]
                )
        Select Distinct
                @DBCode
              , [t].[StockCode]
              , [t].[Lot]
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_Lots-Retesting

```
              , [t].[UnitCost]
        From    ( Select    [StockCode]
                          , [Lot]
                          , [LotRankDescending] = Dense_Rank() Over ( Partition By
[LT].[StockCode] ,
                                                                      [LT].[Lot] Order By
[LT].[TrnDate] Desc, [LT].[TrnType] Asc )
                          , [LT].[TrnDate]
                          , [LT].[UnitCost]
                  From      [LotTransactions] [LT]
                  where TrnType<>''E''
                ) [t]
        Where   [t].[LotRankDescending] = 1;
            End
    End';
--Enable this function to check script changes (try to run script directly against
db manually)

--Print @SQL


--execute script against each db, populating the base tables

        Exec [Process].[ExecForEachDB] @cmd = @SQL1;
        Exec [Process].[ExecForEachDB] @cmd = @SQL2;
        Exec [Process].[ExecForEachDB] @cmd = @SQL3;
        Exec [Process].[ExecForEachDB] @cmd = @SQL4;
        Exec [Process].[ExecForEachDB] @cmd = @SQLLatestLots;

--define the results you want to return

        Create Table [#Results]
            (
              [DatabaseName] Varchar(150) Collate Latin1_General_BIN
            , [StockCode] Varchar(35) Collate Latin1_General_BIN
            , [Lot] Varchar(35) Collate Latin1_General_BIN
            , [Bin] Varchar(10) Collate Latin1_General_BIN
            , [QtyOnHand] Numeric(20 , 8)
            , [ExpiryDate] DateTime2
            , [Description] Varchar(150) Collate Latin1_General_BIN
            , [LotNumber] Varchar(150) Collate Latin1_General_BIN
            , [Warehouse] Varchar(150) Collate Latin1_General_BIN
            , [UnitCost] Numeric(20 , 8)
            );

--Placeholder to create indexes as required


--script to combine base data and insert into results table

        Insert  [#Results]
                ( [DatabaseName]
                , [StockCode]
                , [Lot]
                , [Bin]
                , [QtyOnHand]
                , [ExpiryDate]
                , [Description]
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_Lots-Retesting

```
                , [LotNumber]
                , [Warehouse]
                , [UnitCost]
                )
                Select  [LD].[DatabaseName]
                      , [LD].[StockCode]
                      , [LD].[Lot]
                      , [LD].[Bin]
                      , [LD].[QtyOnHand]
                      , [LD].[ExpiryDate]
                      , [im].[Description]
                      , [LotNumber] = [l].[JobPurchOrder]
                      , [Warehouse] = [IWC].[Description]
                      , [LL].[UnitCost]
                From    [#LotDetail] [LD]
                        Inner Join [#InvWhControl] [IWC] On [LD].[Warehouse] =
[IWC].[Warehouse]
                                                            And [LD].[DatabaseName]
= [IWC].[DatabaseName]
                        Left Join [#InvMaster] As [im] On [im].[StockCode] =
[LD].[StockCode]
                                                          And [im].[DatabaseName] =
[LD].[DatabaseName]
                        Left Outer Join [#Lots] As [l] On [l].[Lot] = [LD].[Lot]
                                                          And [l].[DatabaseName] =
[LD].[DatabaseName]
                        Left Join [#LatestLots] As [LL] On [LD].[Lot] = [LL].[Lot]
                                                           And [LL].[StockCode] =
[LD].[StockCode]
                                                           And [LL].[DatabaseName] =
[LD].[DatabaseName]
                Where   ( [LD].[QtyOnHand] > 0 )
                Order By [IWC].[Description]
                      , [LD].[StockCode];


--return results

        Select  [Company] = [cn].[CompanyName]
              , [R].[StockCode]
              , [R].[Lot]
              , [R].[Bin]
              , [R].[QtyOnHand]
              , [ExpiryDate] = Cast([R].[ExpiryDate] As Date)
              , [R].[Description]
              , [R].[LotNumber]
              , [R].[Warehouse]
              , [R].[UnitCost]
        From    [#Results] [R]
                Left Join [BlackBox].[Lookups].[CompanyNames] As [cn] On
[cn].[Company] = [R].[DatabaseName];

        Drop Table [#InvMaster];
        Drop Table [#InvWhControl];
        Drop Table [#LotDetail];
        Drop Table [#Lots];
        Drop Table [#Results];
        Drop Table [#LatestLots];
    End;
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_Lots-Retesting

```
GO
EXEC sp_addextendedproperty N'MS_Description', N'list of lots and when they need to be retested', 'SCHEMA', N'Report', 'PROCEDURE', N'UspResults_LotsRetesting', NULL, NULL
GO
```

Uses

[Lookups].[CompanyNames]
[Process].[ExecForEachDB]
[Process].[UspInsert_RedTagLogs]
[Report]

Author: Johnson, Chris

[Report].[UspResults_LotTraceability]

MS_Description

searches for all lot transactions

Parameters

Name	Data Type	Max Length (Bytes)
@Company	varchar(max)	max
@RedTagType	char	1
@RedTagUse	varchar(500)	500

SQL Script

```
CREATE Proc [Report].[UspResults_LotTraceability]
    (
      @Company Varchar(Max)
    , @RedTagType Char(1)
    , @RedTagUse Varchar(500)
    )
As
    Begin
/*
Template designed by Chris Johnson, Prometic Group September 2015
Stored procedure set out to query multiple databases with the same information and
return it in a collated format
*/
        Set NoCount On;

        If IsNumeric(@Company) = 0
            Begin
                Select  @Company = Upper(@Company);
            End;

--Red tag

        Declare @RedTagDB Varchar(255)= Db_Name();
        Exec [Process].[UspInsert_RedTagLogs] @StoredProcDb = 'BlackBox' ,
            @StoredProcSchema = 'Report' ,
            @StoredProcName = 'UspResults_LotTraceability' ,
            @UsedByType = @RedTagType , @UsedByName = @RedTagUse ,
            @UsedByDb = @RedTagDB;

--remove nocount on to speed up query

        Set NoCount On;



--list the tables that are to be pulled back from each DB - if they are not found
the script will not be run against that db
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_Lot-Traceability

```
        Declare @ListOfTables Varchar(Max) = 'MdnMasterRep,WipMaster,CusSorMaster+';

--create temporary tables to be pulled from different databases, including a column to id

--[#InvInspect][#LotTransactions][#InvMaster]

        Create Table [#InvInspect]
            (
              [DatabaseName] Varchar(150)            collate latin1_general_bin
            , [InspNarration] Varchar(100)            collate latin1_general_bin
            , [Lot] Varchar(50)                       collate latin1_general_bin
            , [GrnReceiptDate] Date
            , [StockCode] Varchar(30)                 collate latin1_general_bin
            , [TotalReceiptQty] Numeric(20 , 8)
            );
        Create Table [#LotTransactions]
            (
              [DatabaseName] Varchar(150)             collate latin1_general_bin
            , [Lot] Varchar(50)                       collate latin1_general_bin
            , [StockCode] Varchar(30)                 collate latin1_general_bin
            , [TrnDate] Date
            , [Warehouse] Varchar(10)                 collate latin1_general_bin
            , [Bin] Varchar(20)                       collate latin1_general_bin
            , [Job] Varchar(20)                       collate latin1_general_bin
            , [Reference] Varchar(30)                 collate latin1_general_bin
            , [UnitCost] Numeric(20 , 2)
            , [TrnQuantity] Numeric(20 , 8)
            , [TrnValue] Numeric(20 , 8)
            , [TrnType] Char(1)                       collate latin1_general_bin
            , [JobPurchOrder] Varchar(20)             collate latin1_general_bin
            , [Customer] Varchar(15)                  collate latin1_general_bin
            );
        Create Table [#InvMaster]
            (
              [DatabaseName] Varchar(150)             collate latin1_general_bin
            , [Description] Varchar(50)               collate latin1_general_bin
            , [StockUom] Varchar(20)                  collate latin1_general_bin
            , [StockCode] Varchar(30)                 collate latin1_general_bin
            );
        Create Table [#ArCustomer]
            (
              [DatabaseName] Varchar(150)             collate latin1_general_bin
            , [Customer] Varchar(15)                  collate latin1_general_bin
            , [Name] Varchar(50)                      collate latin1_general_bin
            );
        Create Table [#WipMaster]
            (
              [DatabaseName] Varchar(150)             collate latin1_general_bin
            , [Job] Varchar(20)                       collate latin1_general_bin
            , [ActCompleteDate] Date
            );



--create script to pull data from each db into the tables
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_Lot-Traceability

```
        Declare @SQLInvInspect Varchar(Max) = 'USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            BEGIN
                Insert [#InvInspect]
            ( [DatabaseName]
            , [InspNarration]
            , [Lot]
            , [GrnReceiptDate]
            , [StockCode]
            , [TotalReceiptQty]
            )
                SELECT @DBCode
            , InspNarration = Case When [II].[InspNarration] = ''''
                                Then Null
                                Else [II].[InspNarration]
                            End
            , [II].[Lot]
            , [II].[GrnReceiptDate]
            , [II].[StockCode]
            , [II].[TotalReceiptQty] FROM [InvInspect] As [II]
            End
    End';
        Declare @SQLLotTransactions Varchar(Max) = 'USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            BEGIN
                Insert [#LotTransactions]
                ( [DatabaseName]
                , [Lot]
                , [StockCode]
                , [TrnDate]
                , [Warehouse]
                , [Bin]
                , [Job]
                , [Reference]
                , [UnitCost]
                , [TrnQuantity]
                , [TrnValue]
                , TrnType
                , [JobPurchOrder]
                , Customer
                )
                SELECT @DBCode
                     , [LT].[Lot]
                 , [LT].[StockCode]
```

Author: Johnson, Chris

```
                , [LT].[TrnDate]
                , [LT].[Warehouse]
                , [LT].[Bin]
                , [Job] = Case When [LT].[Job] = '''' Then Null
                               Else [LT].[Job]
                          End
                , [Reference] = Case When [LT].[Reference] = '''' Then Null
                                     Else [LT].[Reference]
                                End
                , [LT].[UnitCost]
                , [LT].[TrnQuantity]
                , [LT].[TrnValue]
                , [LT].TrnType
                , [LT].[JobPurchOrder]
                , [LT].Customer FROM [LotTransactions] As [LT]
            End
    End';
        Declare @SQLInvMaster Varchar(Max) = 'USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            BEGIN
                Insert [#InvMaster]
                ( [DatabaseName]
                , [Description]
                , [StockUom]
                , [StockCode]
                )
                SELECT @DBCode
                , [IM].[Description]
                , [IM].[StockUom]
                , [IM].[StockCode] FROM [InvMaster] As [IM]
            End
    End';
        Declare @SQLArCustomer Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            BEGIN
                Insert  [#ArCustomer]
                        ( [DatabaseName]
                        , [Customer]
                        , [Name]
                        )
                Select  @DBCode
                      , [AC].[Customer]
                      , [AC].[Name]
                From    [ArCustomer] As [AC];
```

Author: Johnson, Chris

```
            End
    End';
        Declare @SQLWipMaster Varchar(Max) = 'USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            BEGIN
                Insert [#WipMaster]
                    ( [DatabaseName]
                    , [Job]
                    , [ActCompleteDate]
                    )
                Select  @DBCode
                    ,[WM].[Job]
                    ,[WM].[ActCompleteDate]
                    From    [WipMaster] [WM];
            End
    End';
--Enable this function to check script changes (try to run script directly against
db manually)

--Print @SQL



--execute script against each db, populating the base tables

        Exec [Process].[ExecForEachDB_WithTableCheck] @cmd = @SQLInvInspect , --
nvarchar(max)

            @SchemaTablesToCheck = @ListOfTables;
        Exec [Process].[ExecForEachDB_WithTableCheck] @cmd = @SQLInvMaster , --
nvarchar(max)

            @SchemaTablesToCheck = @ListOfTables;
        Exec [Process].[ExecForEachDB_WithTableCheck] @cmd = @SQLLotTransactions , -
- nvarchar(max)

            @SchemaTablesToCheck = @ListOfTables;
        Exec [Process].[ExecForEachDB_WithTableCheck] @cmd = @SQLArCustomer , --
nvarchar(max)

            @SchemaTablesToCheck = @ListOfTables;
        Exec [Process].[ExecForEachDB_WithTableCheck] @cmd = @SQLWipMaster ,
            @SchemaTablesToCheck = @ListOfTables;



--define the results you want to return

        Create Table [#Results]
            (
              [Company] Varchar(300)                collate latin1_general_bin
            , [SupplierLotNumber] Varchar(100)        collate latin1_general_bin
            , [Lot] Varchar(50)                        collate latin1_general_bin
            , [GrnReceiptDate] Date
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_Lot-Traceability

```
            , [StockCode] Varchar(30)                collate latin1_general_bin
            , [StockDescription] Varchar(50)        collate latin1_general_bin
            , [TrnDate] Date
            , [Warehouse] Varchar(10)                collate latin1_general_bin
            , [Bin] Varchar(20)                        collate latin1_general_bin
            , [Job] Varchar(20)                        collate latin1_general_bin
            , [TrnTypeDescription] Varchar(100)        collate latin1_general_bin
            , [Reference] Varchar(30)                collate latin1_general_bin
            , [TotalReceiptQty] Numeric(20 , 8)
            , [StockUom] Varchar(20)                collate latin1_general_bin
            , [UnitCost] Numeric(20 , 2)
            , [TrnQuantity] Numeric(20 , 8)
            , [TrnValue] Numeric(20 , 2)
            , [MasterJob] Varchar(30)                collate latin1_general_bin
            , [CustomerName] Varchar(50)            collate latin1_general_bin
            , [MasterJobDate] Date
            );

--Placeholder to create indexes as required

        Create Table [#LotMasterJob]
            (
              [Lot] Varchar(50)    Collate Latin1_General_BIN
            , [MasterJob] Varchar(30) Collate Latin1_General_BIN
            , [TempLot] Varchar(50) Collate Latin1_General_BIN
            , [ActCompleteDate] Date
            );

        Insert  [#LotMasterJob]
                ( [Lot]
                , [MasterJob]
                , [ActCompleteDate]
                )
                Select Distinct
                        [LT].[Lot]
                      , [LT].[JobPurchOrder]
                      , [WM].[ActCompleteDate]
                From    [#LotTransactions] As [LT]
                        Left Join [#WipMaster] [WM]
                            On [WM].[DatabaseName] = [LT].[DatabaseName]
                               And [WM].[Job] = [LT].[JobPurchOrder]
                Where   [LT].[TrnType] = 'R'
                        And [LT].[Reference] <> '';

        Create Index [LMJ] On [#LotMasterJob] ([Lot]);

        --get the lot that was issued to a kitting job

        Update  [#LotMasterJob]
        Set     [TempLot] = [LT].[Lot]
        From    [#LotMasterJob] [LMJ]
                Left Join [#LotTransactions] [LT]
                    On [LMJ].[MasterJob] = [LT].[Job]
        Where   [LMJ].[MasterJob] Like 'KN%'
                And [LT].[TrnType] = 'I';

        --use the lot that was issued to update the master job
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_Lot-Traceability

```
        Update  [LMJ]
        Set     [LMJ].[MasterJob] = [LMJ2].[MasterJob]
        From    [#LotMasterJob] [LMJ]
                Left Join [#LotMasterJob] [LMJ2]
                    On [LMJ2].[Lot] = [LMJ].[TempLot]
        Where   [LMJ].[TempLot] Is Not Null;


--script to combine base data and insert into results table

        Insert  [#Results]
                ( [Company]
                , [SupplierLotNumber]
                , [Lot]
                , [GrnReceiptDate]
                , [StockCode]
                , [StockDescription]
                , [TrnDate]
                , [Warehouse]
                , [Bin]
                , [Job]
                , [TrnTypeDescription]
                , [Reference]
                , [TotalReceiptQty]
                , [StockUom]
                , [UnitCost]
                , [TrnQuantity]
                , [TrnValue]
                , [MasterJob]
                , [CustomerName]
                , [MasterJobDate]
                )
                Select  [Company] = Coalesce([II].[DatabaseName] ,
                                             [LT].[DatabaseName])
                      , [SupplierLotNumber] = [II].[InspNarration]
                      , [Lot] = Case When IsNumeric(Coalesce([II].[Lot] ,
                                                             [LT].[Lot])) = 1
                                     Then Convert(Varchar(50) , Convert(Int ,
Coalesce([II].[Lot] ,
                                                                        [LT].[Lot])))
                                     Else Coalesce([II].[Lot] , [LT].[Lot])
                                End
                      , [II].[GrnReceiptDate]
                      , [StockCode] = Coalesce([II].[StockCode] ,
                                               [LT].[StockCode])
                      , [StockDescription] = [IM].[Description]
                      , [TrnDate] = [LT].[TrnDate]
                      , [LT].[Warehouse]
                      , [LT].[Bin]
                      , [Job] = [LT].[Job]
                      , [TransactionType] = [LTT].[TrnTypeDescription]
                      , [Reference] = [LT].[Reference]
                      , [II].[TotalReceiptQty]
                      , [IM].[StockUom]
                      , [LT].[UnitCost]
                      , [TrnQuantity] = [LT].[TrnQuantity]
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_Lot-Traceability

```
                        * [TTAM].[AmountModifier]
                    , [LT].[TrnValue]
                    , [MasterJob] = Case When IsNumeric([LMJ].[MasterJob]) = 1
                                        Then Convert(Varchar(30) , Convert(Int ,
[LMJ].[MasterJob]))
                                        Else [LMJ].[MasterJob]
                                    End
                    , [CustomerName] = [AC].[Name]
                    , [MasterJobDate] = [LMJ].[ActCompleteDate]
               From   [#InvInspect] As [II]
                      Full Outer Join [#LotTransactions] As [LT]
                          On [LT].[Lot] = [II].[Lot]
                                                          --And [LT].[StockCode]
= [II].[StockCode]

                             And [LT].[DatabaseName] = [II].[DatabaseName]
                      Left Join [#InvMaster] As [IM]
                          On Coalesce([II].[StockCode] , [LT].[StockCode]) =
[IM].[StockCode]
                             And Coalesce([II].[DatabaseName] ,
                                          [LT].[DatabaseName]) = [IM].[Database-
Name]
                      Left Join [BlackBox].[Lookups].[LotTransactionTrnType] [LTT]
                          On [LT].[TrnType] = [LTT].[TrnType]
                      Left Join [Lookups].[TrnTypeAmountModifier] [TTAM]
                          On [LT].[DatabaseName] = [TTAM].[Company]
                             And [TTAM].[TrnType] = [LT].[TrnType]
                      Left Join [#LotMasterJob] As [LMJ]
                          On [LMJ].[Lot] = [LT].[Lot]
                      Left Join [#ArCustomer] As [AC]
                          On [AC].[Customer] = [LT].[Customer]
                             And [AC].[DatabaseName] = [LT].[DatabaseName]
               Where  Coalesce([II].[Lot] , [LT].[Lot] , '') <> ''
               Order By [SupplierLotNumber] Desc
                     , Case When IsNumeric(Coalesce([II].[Lot] , [LT].[Lot])) = 1
                            Then Convert(Varchar(50) , Convert(Int ,
Coalesce([II].[Lot] ,
                                                          [LT].[Lot])))
                            Else Coalesce([II].[Lot] , [LT].[Lot])
                      End Asc
                    , [LT].[TrnDate];


--return results


        Set NoCount Off;
        Select  [CN].[CompanyName]
              , [R].[Company]
              , [R].[SupplierLotNumber]
              , [R].[Lot]
              , [R].[GrnReceiptDate]
              , [R].[StockCode]
              , [R].[StockDescription]
              , [R].[TrnDate]
              , [Warehouse] = [W].[WarehouseDescription]
              , [R].[Bin]
              , [R].[Job]
```

Author: Johnson, Chris

```
              , [TransactionType] = [R].[TrnTypeDescription]
              , [R].[Reference]
              , [R].[TotalReceiptQty]
              , [R].[StockUom]
              , [R].[UnitCost]
              , [R].[TrnQuantity]
              , [R].[TrnValue]
              , [R].[MasterJob]
              , [R].[CustomerName]
              , [R].[MasterJobDate]
        From    [#Results] As [R]
                Left Join [Lookups].[CompanyNames] As [CN]
                    On [CN].[Company] = [R].[Company]
                Left Join [Lookups].[Warehouse] As [W]
                    On [W].[Warehouse] = [R].[Warehouse]
                       And [W].[Company] = [R].[Company];

    End;
GO
EXEC sp_addextendedproperty N'MS_Description', N'searches for all lot transactions',
'SCHEMA', N'Report', 'PROCEDURE', N'UspResults_LotTraceability', NULL, NULL
GO
```

Uses

[Lookups].[CompanyNames]
[Lookups].[LotTransactionTrnType]
[Lookups].[TrnTypeAmountModifier]
[Lookups].[Warehouse]
[Process].[ExecForEachDB_WithTableCheck]
[Process].[UspInsert_RedTagLogs]
[Report]

[Report].[UspResults_LotTraceability_WithRuntimes]

MS_Description

searches for all lot transactions

Parameters

Name	Data Type	Max Length (Bytes)
@Company	varchar(max)	max
@RedTagType	char	1
@RedTagUse	varchar(500)	500

SQL Script

```
CREATE Proc [Report].[UspResults_LotTraceability_WithRuntimes]
    (
      @Company Varchar(Max)
    , @RedTagType Char(1)
    , @RedTagUse Varchar(500)
    )
As
    Begin
        Set NoCount On;

        If IsNumeric(@Company) = 0
            Begin
                Select  @Company = Upper(@Company);
            End;

--Red tag

        Declare @RedTagDB Varchar(255)= Db_Name();
        Exec [Process].[UspInsert_RedTagLogs] @StoredProcDb = 'BlackBox' ,
            @StoredProcSchema = 'Report' ,
            @StoredProcName = 'UspResults_LotTraceability_WithRuntimes' ,
            @UsedByType = @RedTagType , @UsedByName = @RedTagUse ,
            @UsedByDb = @RedTagDB;


--list the tables that are to be pulled back from each DB - if they are not found
the script will not be run against that db

        Declare @ListOfTables Varchar(Max) = 'InvInspect,LotTransactions,Inv-
Master,ArCustomer,WipLabJnl';

--create temporary tables to be pulled from different databases, including a column
to id

        Create Table [#InvInspect]
            (
              [DatabaseName] Varchar(150)            collate latin1_general_bin
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_Lot-Traceability_WithRuntimes

```
            , [InspNarration] Varchar(100)            collate latin1_general_bin
            , [Lot] Varchar(50)                       collate latin1_general_bin
            , [GrnReceiptDate] Date
            , [StockCode] Varchar(30)                 collate latin1_general_bin
            , [TotalReceiptQty] Numeric(20 , 8)
            );
        Create Table [#LotTransactions]
            (
              [DatabaseName] Varchar(150)             collate latin1_general_bin
            , [Lot] Varchar(50)                       collate latin1_general_bin
            , [StockCode] Varchar(30)                 collate latin1_general_bin
            , [TrnDate] Date
            , [Warehouse] Varchar(10)                 collate latin1_general_bin
            , [Bin] Varchar(20)                       collate latin1_general_bin
            , [Job] Varchar(20)                       collate latin1_general_bin
            , [Reference] Varchar(30)                 collate latin1_general_bin
            , [UnitCost] Numeric(20 , 2)
            , [TrnQuantity] Numeric(20 , 8)
            , [TrnValue] Numeric(20 , 8)
            , [TrnType] Char(1)                       collate latin1_general_bin
            , [JobPurchOrder] Varchar(20)             collate latin1_general_bin
            , [Customer] Varchar(15)                  collate latin1_general_bin
            , [Source] Char(1)                        collate latin1_general_bin
            , [SalesOrder] Varchar(20)                collate latin1_general_bin
            , [Bucket] Int
            );
        Create Table [#InvMaster]
            (
              [DatabaseName] Varchar(150)             collate latin1_general_bin
            , [Description] Varchar(50)               collate latin1_general_bin
            , [StockUom] Varchar(20)                  collate latin1_general_bin
            , [StockCode] Varchar(30)                 collate latin1_general_bin
            );
        Create Table [#ArCustomer]
            (
              [DatabaseName] Varchar(150)             collate latin1_general_bin
            , [Customer] Varchar(15)                  collate latin1_general_bin
            , [Name] Varchar(50)                      collate latin1_general_bin
            );
        Create Table [#WipLabJnl]
            (
              [DatabaseName] Varchar(150)             collate latin1_general_bin
            , [RunTime] Numeric(20 , 6)
            , [Job] Varchar(20)                       collate latin1_general_bin
            );
        Create Table [#Labour]
            (
              [Lot] Varchar(50)                       collate latin1_general_bin
            , [JobPurchOrder] Varchar(20)             collate latin1_general_bin
            , [RunTime] Numeric(20 , 5)
            , [DatabaseName] Varchar(150)             collate latin1_general_bin
            , [TotalLots] Int
            );

--create script to pull data from each db into the tables

        Declare @SQLInvInspect Varchar(Max) = 'USE [?];
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_Lot-Traceability_WithRuntimes

```
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company)
            + ''' = ''ALL''
            BEGIN
                    Insert [#InvInspect]
                            ( [DatabaseName]
                            , [InspNarration]
                            , [Lot]
                            , [GrnReceiptDate]
                            , [StockCode]
                            , [TotalReceiptQty]
                            )
                    SELECT @DBCode
                            , InspNarration = Case When [II].[InspNarration] = ''''
                                                  Then Null
                                                  Else [II].[InspNarration]
                                                  End
                            , [II].[Lot]
                            , [II].[GrnReceiptDate]
                            , [II].[StockCode]
                            , [II].[TotalReceiptQty]
                    FROM [InvInspect] As [II]
            End
    End';
        Declare @SQLLotTransactions Varchar(Max) = 'USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company)
            + ''' = ''ALL''
            BEGIN
                Insert [#LotTransactions]
                    ( [DatabaseName]
                    , [Lot]
                    , [StockCode]
                    , [TrnDate]
                    , [Warehouse]
                    , [Bin]
                    , [Job]
                    , [Reference]
                    , [UnitCost]
                    , [TrnQuantity]
                    , [TrnValue]
                    , TrnType
                    , [JobPurchOrder]
                    , [Customer]
                    , [Source]
                    , [SalesOrder]
                    , [Bucket]
```

Author: Johnson, Chris

```
                    )
                SELECT
                    @DBCode
                    , [LT].[Lot]
                    , [LT].[StockCode]
                    , [LT].[TrnDate]
                    , [LT].[Warehouse]
                    , [LT].[Bin]
                    , [Job] = Case When [LT].[Job] = '''' Then Null
                              Else [LT].[Job]
                              End
                    , [Reference] = Case When [LT].[Reference] = '''' Then Null
                              Else [LT].[Reference]
                              End
                    , [LT].[UnitCost]
                    , [LT].[TrnQuantity]
                    , [LT].[TrnValue]
                    , [LT].TrnType
                    , [LT].[JobPurchOrder]
                    , [LT].Customer
                    , [LT].[Source]
                    , [LT].[SalesOrder]
                    , [LT].[Bucket]
                FROM [LotTransactions] As [LT]
            End
    End';
        Declare @SQLInvMaster Varchar(Max) = 'USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            BEGIN
                Insert [#InvMaster]
                        ( [DatabaseName]
                        , [Description]
                        , [StockUom]
                        , [StockCode]
                        )
                SELECT
                        @DBCode
                        , [IM].[Description]
                        , [IM].[StockUom]
                        , [IM].[StockCode]
                FROM [InvMaster] As [IM]
            End
    End';
        Declare @SQLArCustomer Varchar(Max) = 'USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
```

Author: Johnson, Chris

```
            BEGIN
                Insert  [#ArCustomer]
                        ( [DatabaseName]
                        , [Customer]
                        , [Name]
                        )
                Select  @DBCode
                      , [AC].[Customer]
                      , [AC].[Name]
                From    [ArCustomer] As [AC];
            End
    End';
        Declare @SQLWipLabJnl Varchar(Max) = 'USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            BEGIN
                Insert [#WipLabJnl]
                        ( [DatabaseName] , [RunTime] , [Job] )
                SELECT @DBCode
                     , [WLJ].[RunTime]
                     , [WLJ].[Job]
                FROM [WipLabJnl] [WLJ]
            End
    End';

--execute script against each db, populating the base tables

        Exec [Process].[ExecForEachDB_WithTableCheck] @cmd = @SQLInvInspect ,
            @SchemaTablesToCheck = @ListOfTables;
        Exec [Process].[ExecForEachDB_WithTableCheck] @cmd = @SQLInvMaster ,
            @SchemaTablesToCheck = @ListOfTables;
        Exec [Process].[ExecForEachDB_WithTableCheck] @cmd = @SQLLotTransactions ,
            @SchemaTablesToCheck = @ListOfTables;
        Exec [Process].[ExecForEachDB_WithTableCheck] @cmd = @SQLArCustomer ,
            @SchemaTablesToCheck = @ListOfTables;
        Exec [Process].[ExecForEachDB_WithTableCheck] @cmd = @SQLWipLabJnl ,
            @SchemaTablesToCheck = @ListOfTables;

--define the results to return

        Create Table [#Results]
            (
              [Company] Varchar(300)                collate latin1_general_bin
            , [SupplierLotNumber] Varchar(100)        collate latin1_general_bin
            , [Lot] Varchar(50)                       collate latin1_general_bin
            , [GrnReceiptDate] Date
            , [StockCode] Varchar(30)                 collate latin1_general_bin
            , [StockDescription] Varchar(50)          collate latin1_general_bin
            , [TrnDate] Date
            , [Warehouse] Varchar(10)                 collate latin1_general_bin
            , [Bin] Varchar(20)                       collate latin1_general_bin
            , [Job] Varchar(20)                       collate latin1_general_bin
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_Lot-Traceability_WithRuntimes

```
            , [TrnTypeDescription] Varchar(100)        collate latin1_general_bin
            , [Reference] Varchar(30)                  collate latin1_general_bin
            , [TotalReceiptQty] Numeric(20 , 8)
            , [StockUom] Varchar(20)                   collate latin1_general_bin
            , [UnitCost] Numeric(20 , 2)
            , [TrnQuantity] Numeric(20 , 8)
            , [TrnValue] Numeric(20 , 2)
            , [MasterJob] Varchar(30)                  collate latin1_general_bin
            , [CustomerName] Varchar(50)               collate latin1_general_bin
            , [TranRank] Int
            );

--Placeholder to create indexes as required

        Create Table [#LotMasterJob]
            (
              [Lot] Varchar(50)                        collate latin1_general_bin
            , [MasterJob] Varchar(30)                  collate latin1_general_bin
            , [TempLot] Varchar(50)                    collate latin1_general_bin
            , [DatabaseName] Varchar(150)              collate latin1_general_bin
            );

        Insert  [#LotMasterJob]
                ( [Lot]
                , [MasterJob]
                , [DatabaseName]
                )
                Select Distinct
                        [LT].[Lot]
                      , [LT].[JobPurchOrder]
                      , [LT].[DatabaseName]
                From    [#LotTransactions] As [LT]
                Where   [LT].[TrnType] = 'R'
                        And [LT].[Reference] <> '';

        Create Index [LMJ] On [#LotMasterJob] ([Lot]);

        --get the lot that was issued to a kitting job

        Update  [#LotMasterJob]
        Set     [TempLot] = [LT].[Lot]
        From    [#LotMasterJob] [LMJ]
                Left Join [#LotTransactions] [LT]
                    On [LMJ].[MasterJob] = [LT].[Job]
        Where   [LMJ].[MasterJob] Like 'KN%'
                And [LT].[TrnType] = 'I';

        --use the lot that was issued to update the master job

        Update  [LMJ]
        Set     [LMJ].[MasterJob] = [LMJ2].[MasterJob]
        From    [#LotMasterJob] [LMJ]
                Left Join [#LotMasterJob] [LMJ2]
                    On [LMJ2].[Lot] = [LMJ].[TempLot]
        Where   [LMJ].[TempLot] Is Not Null;

        --List of work
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_Lot-Traceability_WithRuntimes

```
        Insert  [#Labour]
                ( [Lot]
                , [JobPurchOrder]
                , [RunTime]
                , [DatabaseName]
                )
                Select  [Lot] = Case When IsNumeric([lt].[Lot]) = 1
                                    Then Convert(Varchar(50) , Convert(Int ,
[lt].[Lot]))
                                    Else [lt].[Lot]
                               End
                      , [lt].[JobPurchOrder]
                      , [RunTime] = Sum([wlj].[RunTime])
                      , [lt].[DatabaseName]
                From    ( Select Distinct
                                 [Lot]
                               , [JobPurchOrder]
                               , [DatabaseName]
                          From   [#LotTransactions]
                          Where  [Source] = 'J'
                        ) As [lt]
                        Left Join [#WipLabJnl] As [wlj]
                            On [wlj].[Job] = [lt].[JobPurchOrder]
                               And [wlj].[DatabaseName] = [lt].[DatabaseName]
                Group By [lt].[Lot]
                      , [lt].[JobPurchOrder]
                      , [lt].[DatabaseName];

        Update  [#Labour]
        Set     [TotalLots] = [TL]
        From    [#Labour] As [l2]
                Left Join ( Select  [l].[JobPurchOrder]
                                  , [l].[DatabaseName]
                                  , [TL] = Count(Distinct [l].[Lot])
                            From    [#Labour] As [l]
                            Group By [l].[JobPurchOrder]
                                  , [l].[DatabaseName]
                          ) [t]
                    On [t].[JobPurchOrder] = [l2].[JobPurchOrder]
                       And [t].[DatabaseName] = [l2].[DatabaseName];

--script to combine base data and insert into results table

        Insert  [#Results]
                ( [Company]
                , [SupplierLotNumber]
                , [Lot]
                , [GrnReceiptDate]
                , [StockCode]
                , [StockDescription]
                , [TrnDate]
                , [Warehouse]
                , [Bin]
                , [Job]
                , [TrnTypeDescription]
                , [Reference]
                , [TotalReceiptQty]
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_Lot-Traceability_WithRuntimes

```
                , [StockUom]
                , [UnitCost]
                , [TrnQuantity]
                , [TrnValue]
                , [MasterJob]
                , [CustomerName]
                , [TranRank]
                )
                Select  [Company] = Coalesce([II].[DatabaseName] ,
                                             [LT].[DatabaseName])
                      , [SupplierLotNumber] = [II].[InspNarration]
                      , [Lot] = Case When IsNumeric(Coalesce([II].[Lot] ,
                                                             [LT].[Lot])) = 1
                                     Then Convert(Varchar(50) , Convert(Int ,
Coalesce([II].[Lot] ,
                                                                        [LT].[Lot])))
                                     Else Coalesce([II].[Lot] , [LT].[Lot])
                                End
                      , [II].[GrnReceiptDate]
                      , [StockCode] = Coalesce([II].[StockCode] ,
                                               [LT].[StockCode])
                      , [StockDescription] = [IM].[Description]
                      , [TrnDate] = [LT].[TrnDate]
                      , [LT].[Warehouse]
                      , [LT].[Bin]
                      , [Job] = [LT].[Job]
                      , [TransactionType] = [LTT].[TrnTypeDescription]
                      , [Reference] = [LT].[Reference]
                      , [II].[TotalReceiptQty]
                      , [IM].[StockUom]
                      , [LT].[UnitCost]
                      , [LT].[TrnQuantity] * [TTAM].[AmountModifier]
                      , [LT].[TrnValue]
                      , [MasterJob] = Case When IsNumeric([LMJ].[MasterJob]) = 1
                                           Then Convert(Varchar(30) , Convert(Int ,
[LMJ].[MasterJob]))
                                           Else [LMJ].[MasterJob]
                                      End
                      , [CustomerName] = [AC].[Name]
                      , [TranRank] = Dense_Rank() Over ( Partition By [LT].[Lot]
Order By [LT].[TrnDate] Asc, [LT].[SalesOrder] Asc, [LT].[Bucket] Asc )
                From    [#InvInspect] As [II]
                        Full Outer Join [#LotTransactions] As [LT]
                            On [LT].[Lot] = [II].[Lot]
                               And [LT].[DatabaseName] = [II].[DatabaseName]
                        Left Join [#InvMaster] As [IM]
                            On Coalesce([II].[StockCode] , [LT].[StockCode]) =
[IM].[StockCode]
                               And Coalesce([II].[DatabaseName] ,
                                            [LT].[DatabaseName]) = [IM].[Database-
Name]
                        Left Join [BlackBox].[Lookups].[LotTransactionTrnType] [LTT]
                            On [LT].[TrnType] = [LTT].[TrnType]
                        Left Join [#LotMasterJob] As [LMJ]
                            On [LMJ].[Lot] = [LT].[Lot]
                               And [LMJ].[DatabaseName] = [LT].[DatabaseName]
                        Left Join [#ArCustomer] As [AC]
                            On [AC].[Customer] = [LT].[Customer]
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_Lot-Traceability_WithRuntimes

```
                                And [AC].[DatabaseName] = [LT].[DatabaseName]
                        Left Join [Lookups].[TrnTypeAmountModifier] [TTAM]
                            On [TTAM].[TrnType] = [LT].[TrnType]
                               And [TTAM].[Company] = [LT].[DatabaseName]
                Where   Coalesce([II].[Lot] , [LT].[Lot] , '') <> ''
                Order By [SupplierLotNumber] Desc
                      , Case When IsNumeric(Coalesce([II].[Lot] , [LT].[Lot])) = 1
                             Then Convert(Varchar(50) , Convert(Int ,
Coalesce([II].[Lot] ,
                                                                [LT].[Lot])))
                             Else Coalesce([II].[Lot] , [LT].[Lot])
                        End Asc
                      , [LT].[TrnDate];

        Select  [R].[Company]
              , [R].[MasterJob]
              , [JobLeftQuantity] = Sum([R].[TrnQuantity])
        Into    [#Results2]
        From    [#Results] [R]
        Where [R].[Warehouse]='Finished Goods'
        Group By [R].[Company]
              , [R].[MasterJob];

        Set NoCount Off;
--return results

        Select  [CN].[CompanyName]
              , [CN].[ShortName]
              , [R].[Company]
              , [R].[SupplierLotNumber]
              , [R].[Lot]
              , [R].[GrnReceiptDate]
              , [R].[StockCode]
              , [R].[StockDescription]
              , [R].[TrnDate]
              , [Warehouse] = [W].[WarehouseDescription]
              , [R].[Bin]
              , [R].[Job]
              , [TransactionType] = [R].[TrnTypeDescription]
              , [R].[Reference]
              , [R].[TotalReceiptQty]
              , [R].[StockUom]
              , [R].[UnitCost]
              , [R].[TrnQuantity]
              , [R].[TrnValue]
              , [R].[MasterJob]
              , [R].[CustomerName]
              , [RunTime] = Coalesce([L].[RunTime] , 0)
              , [R2].[JobLeftQuantity]
        From    [#Results] As [R]
                Left Join [Lookups].[CompanyNames] As [CN]
                    On [CN].[Company] = [R].[Company]
                Left Join [Lookups].[Warehouse] As [W]
                    On [W].[Warehouse] = [R].[Warehouse]
                       And [W].[Company] = [R].[Company]
                Left Join [#Labour] [L]
                    On [L].[JobPurchOrder] = [R].[MasterJob]
```

Author: Johnson, Chris

```
                        And [L].[Lot] = [R].[Lot]
                        And [L].[DatabaseName] = [R].[Company]
                Left Join [#Results2] [R2]
                    On [R2].[Company] = [CN].[Company]
                        And [R2].[MasterJob] = [R].[MasterJob];
    End;
GO
EXEC sp_addextendedproperty N'MS_Description', N'searches for all lot transactions',
'SCHEMA', N'Report', 'PROCEDURE', N'UspResults_LotTraceability_WithRuntimes', NULL,
NULL
GO
```

Uses

[Lookups].[CompanyNames]
[Lookups].[LotTransactionTrnType]
[Lookups].[TrnTypeAmountModifier]
[Lookups].[Warehouse]
[Process].[ExecForEachDB_WithTableCheck]
[Process].[UspInsert_RedTagLogs]
[Report]

[Report].[UspResults_MovementsTrialBalances]

MS_Description

not in use

Parameters

Name	Data Type	Max Length (Bytes)
@RedTagType	char	1
@RedTagUse	varchar(500)	500

SQL Script

```
--'Development > Chris Johnson > GL Balances And Mvmts';


CREATE Proc [Report].[UspResults_MovementsTrialBalances]
    (
      @RedTagType Char(1)
    , @RedTagUse Varchar(500)
    )
As
    Begin
/*
Template designed by Chris Johnson, Prometic Group September 2015
Stored procedure set out to query multiple databases with the same information and
return it in a collated format
*/
        Set NoCount On;

--Red tag

        Declare @RedTagDB Varchar(255)= Db_Name();
        Exec [Process].[UspInsert_RedTagLogs] @StoredProcDb = 'BlackBox' ,
            @StoredProcSchema = 'Report' ,
            @StoredProcName = 'UspResults_MovementsTrialBalances' ,
            @UsedByType = @RedTagType , @UsedByName = @RedTagUse ,
            @UsedByDb = @RedTagDB;
        Create Table [#Movements]
            (
              [Company] Varchar(50)                          collate
latin1_general_bin
            , [ShortName] Varchar(250)                     collate latin1_general_bin
            , [CompanyName] Varchar(250)                 collate latin1_general_bin
            , [Currency] Varchar(10)                     collate latin1_general_bin
            , [GlCode] Varchar(50)                         collate latin1_general_bin
            , [Description] Varchar(150)                 collate latin1_general_bin
            , [GlGroup] Varchar(50)                        collate
latin1_general_bin
            , [Movement] Numeric(20 , 6)
            , [GlPeriod] Int
```

Author: Johnson, Chris

```
            , [GlYear] Int
            , [Source] Varchar(100)                          collate
latin1_general_bin
            , [Journal] Int
            , [ReportIndex1] Varchar(35)               collate latin1_general_bin
            , [ReportIndex2] Varchar(35)               collate latin1_general_bin
            , [AccountType] Varchar(250)               collate latin1_general_bin
            , [Parse1] Varchar(50)                       collate latin1_general_bin
            , [Parse2] Varchar(50)                       collate latin1_general_bin
            , [Parse3] Varchar(50)                       collate latin1_general_bin
            );

        Create Table [#Balances]
            (
              [Company] Varchar(150)                   collate latin1_general_bin
            , [ShortName] Varchar(250)                   collate latin1_general_bin
            , [CompanyName] Varchar(250)               collate latin1_general_bin
            , [Currency] Varchar(10)                   collate latin1_general_bin
            , [GlCode] Varchar(50)                       collate latin1_general_bin
            , [GlDescription] Varchar(150)               collate latin1_general_bin
            , [ReportIndex1] Varchar(50)               collate latin1_general_bin
            , [ReportIndex2] Varchar(50)               collate latin1_general_bin
            , [GlGroup] Varchar(50)                          collate
latin1_general_bin
            , [GlYear] Int
            , [ClosingBalance] Numeric(20 , 6)
            , [Debit] Numeric(20 , 6)
            , [Credit] Numeric(20 , 6)
            , [Period] Int
            , [AccountType] Varchar(250)               collate latin1_general_bin
            , [Parse1] Varchar(50)                       collate latin1_general_bin
            , [Parse2] Varchar(50)                       collate latin1_general_bin
            , [Parse3] Varchar(50)                       collate latin1_general_bin
            );


        Insert  [#Movements]
                ( [Company]
                , [ShortName]
                , [CompanyName]
                , [Currency]
                , [GlCode]
                , [Description]
                , [GlGroup]
                , [Movement]
                , [GlPeriod]
                , [GlYear]
                , [Source]
                , [Journal]
                , [ReportIndex1]
                , [ReportIndex2]
                , [AccountType]
                , [Parse1]
                , [Parse2]
                , [Parse3]
                )
               Exec [Report].[UspResults_GLMovements] @RedTagType = @RedTagType ,
```

```
                    @RedTagUse = @RedTagUse;
 -- varchar(500)


        Insert  [#Balances]
                ( [Company]
                , [ShortName]
                , [CompanyName]
                , [Currency]
                , [GlCode]
                , [GlDescription]
                , [ReportIndex1]
                , [ReportIndex2]
                , [GlGroup]
                , [GlYear]
                , [ClosingBalance]
                , [Debit]
                , [Credit]
                , [Period]
                , [AccountType]
                , [Parse1]
                , [Parse2]
                , [Parse3]
                )
                Exec [Report].[UspResults_TrialBalance] @RedTagType = @RedTagType ,
                    @RedTagUse = @RedTagUse;
 -- varchar(500)*/


        Create NonClustered Index [tdf] On [#Balances] ([Company],[GlCode],[Gl-
Year],[Period]);
        Create NonClustered Index [tfd] On [#Movements] ([Company],[GlCode],[Gl-
Year],[GlPeriod]);

        Set NoCount off
        Select  [B].[Company]
              , [B].[ShortName]
              , [B].[CompanyName]
              , [B].[Currency]
              , [B].[GlCode]
              , [B].[GlDescription]
              , [B].[ReportIndex1]
              , [B].[ReportIndex2]
              , [B].[GlGroup]
              , [B].[GlYear]
              , [B].[ClosingBalance]
              , [B].[Debit]
              , [B].[Credit]
              , [B].[Period]
              , [M].[Movement]
              , [M].[GlPeriod]
              , [M].[Source]
              , [M].[Journal]
              , [B].[AccountType]
              , [B].[Parse1]
              , [B].[Parse2]
              , [B].[Parse3]
```

Author: Johnson, Chris

```
        From    [#Balances] [B]
                Left Join [#Movements] [M]
                    On [M].[Company] = [B].[Company]
                       And [M].[GlCode] = [B].[GlCode]
                       And [M].[GlYear] = [B].[GlYear]
                       And [B].[Period] = [M].[GlPeriod];
    End;
GO
EXEC sp_addextendedproperty N'MS_Description', N'not in use', 'SCHEMA', N'Report',
'PROCEDURE', N'UspResults_MovementsTrialBalances', NULL, NULL
GO
```

Uses

[Process].[UspInsert_RedTagLogs]
[Report].[UspResults_GLMovements]
[Report].[UspResults_TrialBalance]
[Report]

Author: Johnson, Chris

[Report].[UspResults_OpenPurchaseOrderStock]

MS_Description

returns list of open purchase orders with stock details

Parameters

Name	Data Type	Max Length (Bytes)
@Company	varchar(max)	max
@RedTagType	char	1
@RedTagUse	varchar(500)	500

SQL Script

```
CREATE Proc [Report].[UspResults_OpenPurchaseOrderStock]
    (
      @Company Varchar(Max)
    , @RedTagType Char(1)
    , @RedTagUse Varchar(500)
    )
As
    Begin
/*
Template designed by Chris Johnson, Prometic Group September 2015
Stored procedure set out to query multiple databases with the same information and
return it in a collated format
*/
        If IsNumeric(@Company) = 0
            Begin
                Select  @Company = Upper(@Company);
            End;

--remove nocount on to speed up query

        Set NoCount On;

--Red tag

        Declare @RedTagDB Varchar(255)= Db_Name();
        Exec [Process].[UspInsert_RedTagLogs] @StoredProcDb = 'BlackBox' ,
            @StoredProcSchema = 'Report' ,
            @StoredProcName = 'UspResults_OpenPurchaseOrderStock' ,
            @UsedByType = @RedTagType , @UsedByName = @RedTagUse ,
            @UsedByDb = @RedTagDB;

--list the tables that are to be pulled back from each DB - if they are not found
the script will not be run against that db

        Declare @ListOfTables Varchar(Max) = 'AssetDepreciation,TblApTerms';

--create temporary tables to be pulled from different databases, including a column
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-OpenPurchaseOrderStock

```
to id

        Create Table [#PorMasterHdr]
            (
              [DatabaseName] Varchar(150) Collate Latin1_General_BIN
            , [Supplier] Varchar(15) Collate Latin1_General_BIN
            , [PurchaseOrder] Varchar(20) Collate Latin1_General_BIN
            , [OrderDueDate] Date
            , [DeliveryAddr1] Varchar(40) Collate Latin1_General_BIN
            );
        Create Table [#PorMasterDetail]
            (
              [DatabaseName] Varchar(150) Collate Latin1_General_BIN
            , [PurchaseOrder] Varchar(20) Collate Latin1_General_BIN
            , [Line] Int
            , [MSupCatalogue] Varchar(50) Collate Latin1_General_BIN
            , [MStockDes] Varchar(50) Collate Latin1_General_BIN
            , [MStockCode] Varchar(30) Collate Latin1_General_BIN
            , [MLatestDueDate] Date
            , [MCompleteFlag] Char(1) Collate Latin1_General_BIN
            , [MOrderQty] Numeric(20 , 8)
            , [MReceivedQty] Numeric(20 , 8)
            , [MOrderUom] Varchar(10) Collate Latin1_General_BIN
            , [MWarehouse] Varchar(10) Collate Latin1_General_BIN
            );
        Create Table [#ApSupplier]
            (
              [DatabaseName] Varchar(150) Collate Latin1_General_BIN
            , [Supplier] Varchar(15) Collate Latin1_General_BIN
            , [SupplierName] Varchar(50) Collate Latin1_General_BIN
            );



--create script to pull data from each db into the tables

        Declare @SQLPorMasterDetail Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end'
            + --Only query DBs beginning SysProCompany

            '
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN'
            + --only companies selected in main run, or if companies selected then
all

            '
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT'
            + --count number of tables requested (number of commas plus one)
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-OpenPurchaseOrderStock

```
            '
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')'
            + --Count of the tables requested how many exist in the db

            '
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'','')) '
            + --only if the count matches (all the tables exist in the requested db)
then run the script

            '
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
                Insert [#PorMasterDetail]
                        ( [DatabaseName]
                        , [PurchaseOrder]
                        , [Line]
                        , [MSupCatalogue]
                        , [MStockDes]
                        , [MStockCode]
                        , [MLatestDueDate]
                        , [MCompleteFlag]
                        , [MOrderQty]
                        , [MOrderUom]
                        , [MWarehouse]
                        , [MReceivedQty]
                        )
                SELECT [DatabaseName]=@DBCode
                     , [PMD].[PurchaseOrder]
                     , [PMD].[Line]
                     , [PMD].[MSupCatalogue]
                     , [PMD].[MStockDes]
                     , [PMD].[MStockCode]
                     , [PMD].[MLatestDueDate]
                     , [PMD].[MCompleteFlag]
                     , [PMD].[MOrderQty]
                     , [PMD].[MOrderUom]
                     , [PMD].[MWarehouse]
                     , [PMD].[MReceivedQty] FROM [PorMasterDetail] As [PMD]
            End
    End';
        Declare @SQLPorMasterHdr Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end'
            + --Only query DBs beginning SysProCompany

            '
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN'
            + --only companies selected in main run, or if companies selected then
all

            '
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
```

Author: Johnson, Chris

```
            + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT'
            + --count number of tables requested (number of commas plus one)

            '
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')'
            + --Count of the tables requested how many exist in the db

            '
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'','')) '
            + --only if the count matches (all the tables exist in the requested db)
then run the script

            '
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
                Insert [#PorMasterHdr]
                        ( [DatabaseName]
                        , [Supplier]
                        , [PurchaseOrder]
                        , [OrderDueDate]
                        , [DeliveryAddr1]
                        )
                SELECT [DatabaseName]=@DBCode
                     , [PMH].[Supplier]
                     , [PMH].[PurchaseOrder]
                     , [PMH].[OrderDueDate]
                     , [PMH].[DeliveryAddr1] FROM [PorMasterHdr] As [PMH]
                     where [OrderStatus] not in (''*'')
            End
    End';
        Declare @SQLApSupplier Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end'
            + --Only query DBs beginning SysProCompany

            '
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN'
            + --only companies selected in main run, or if companies selected then
all

            '
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT'
            + --count number of tables requested (number of commas plus one)

            '
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-OpenPurchaseOrderStock

```
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')'
            + --Count of the tables requested how many exist in the db

            '
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'','')) '
            + --only if the count matches (all the tables exist in the requested db)
then run the script

            '
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
            Insert [#ApSupplier]
                    ( [DatabaseName]
                    , [Supplier]
                    , [SupplierName]
                    )
            SELECT [DatabaseName]=@DBCode
                 , [AS].[Supplier]
                 , [AS].[SupplierName] FROM [ApSupplier] As [AS]
            End
    End';
--Enable this function to check script changes (try to run script directly against
db manually)

--Print @SQL


--execute script against each db, populating the base tables

        Exec [Process].[ExecForEachDB] @cmd = @SQLApSupplier;
        Exec [Process].[ExecForEachDB] @cmd = @SQLPorMasterDetail;
        Exec [Process].[ExecForEachDB] @cmd = @SQLPorMasterHdr;

--define the results you want to return


--Placeholder to create indexes as required


--script to combine base data and insert into results table




--return results

        Select  [PMH].[Supplier]
              , [AS].[SupplierName]
              , [PurchaseOrder] = Case When IsNumeric([PMH].[PurchaseOrder]) = 1
                                       Then Convert(Varchar(20) , Convert(Int ,
[PMH].[PurchaseOrder]))
                                       Else [PMH].[PurchaseOrder]
                                  End
              , [PMD].[Line]
              , [SupplierCatalogue] = [PMD].[MSupCatalogue]
              , [StockDescription] = [PMD].[MStockDes]
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-OpenPurchaseOrderStock

```
              , [StockCode] = [PMD].[MStockCode]
              , [PMH].[OrderDueDate]
              , [LatestDueDate] = [PMD].[MLatestDueDate]
              , [Overdue] = Case When DateDiff(Day , [PMD].[MLatestDueDate] ,
                                               GetDate()) > 0 Then 'Overdue'
                                End
              , [Complete] = Case When [PMD].[MCompleteFlag] = '' Then 'N'
                                  Else [PMD].[MCompleteFlag]
                                End
              , [PMD].[MOrderQty]
              , [PMD].[MOrderUom]
              , [PMD].[MWarehouse]
              , [PMH].[DeliveryAddr1]
              , [PMD].[MReceivedQty]
        From    [#PorMasterHdr] As [PMH]
                Left Join [#PorMasterDetail] As [PMD] On [PMD].[PurchaseOrder] =
[PMH].[PurchaseOrder]
                                                         And [PMD].[DatabaseName] =
[PMH].[DatabaseName]
                Left Join [#ApSupplier] As [AS] On [AS].[Supplier] =
[PMH].[Supplier]
                                                   And [AS].[DatabaseName] =
[PMH].[DatabaseName]
        Where   Coalesce([PMD].[MCompleteFlag] , 'N') <> 'Y'
                And [PMD].[MStockCode] Not In ( '' , 'N/A' )
                And [PMD].[MWarehouse] = 'RM'
        Order By [AS].[SupplierName] Asc;


    End;

GO
EXEC sp_addextendedproperty N'MS_Description', N'returns list of open purchase
orders with stock details', 'SCHEMA', N'Report', 'PROCEDURE', N'UspResults_Open-
PurchaseOrderStock', NULL, NULL
GO
```

Uses

[Process].[ExecForEachDB]
[Process].[UspInsert_RedTagLogs]
[Report]

Author: Johnson, Chris

[Report].[UspResults_OutstandingPurchaseOrders]

MS_Description

list of purchase orders outstanding

Parameters

Name	Data Type	Max Length (Bytes)
@Company	varchar(max)	max
@RedTagType	char	1
@RedTagUse	varchar(500)	500

SQL Script

```
CREATE Proc [Report].[UspResults_OutstandingPurchaseOrders]
    (
      @Company Varchar(Max)
    , @RedTagType Char(1)
    , @RedTagUse Varchar(500)
    )
As
    Begin
/*
Template designed by Chris Johnson, Prometic Group September 2015
Stored procedure set out to bring back purchase orders which do not have receipts
*/
        If IsNumeric(@Company) = 0
            Begin
                Select  @Company = Upper(@Company);
            End;

--remove nocount on to speed up query

        Set NoCount On;

--Red tag

        Declare @RedTagDB Varchar(255)= Db_Name();
        Exec [Process].[UspInsert_RedTagLogs] @StoredProcDb = 'BlackBox' ,
            @StoredProcSchema = 'Report' ,
            @StoredProcName = 'UspResults_OutstandingPurchaseOrders' ,
            @UsedByType = @RedTagType , @UsedByName = @RedTagUse ,
            @UsedByDb = @RedTagDB;

--list the tables that are to be pulled back from each DB - if they are not found
the script will not be run against that db

        Declare @ListOfTables Varchar(Max) = 'PorMasterHdr,ApSupplier,PorMaster-
Detail';

--create temporary tables to be pulled from different databases, including a column
```

Author: Johnson, Chris

```
to id

        Create Table [#PorMasterHdr]
            (
              [DatabaseName] Varchar(150)    Collate Latin1_General_BIN
            , [PurchaseOrder] Varchar(35)    Collate Latin1_General_BIN
            , [OrderEntryDate] DateTime2
            , [OrderDueDate] DateTime2
            , [OrderStatus] Varchar(35)        Collate Latin1_General_BIN
            , [Supplier] Varchar(15)        Collate Latin1_General_BIN
            , [CancelledFlag] Char(1)        Collate Latin1_General_BIN
            );
        Create Table [#ApSupplier]
            (
              [DatabaseName] Varchar(150)    Collate Latin1_General_BIN
            , [Supplier] Varchar(15)        Collate Latin1_General_BIN
            , [SupplierName] Varchar(150)    Collate Latin1_General_BIN
            );
        Create Table [#PorMasterDetail]
            (
              [DatabaseName] Varchar(150)    Collate Latin1_General_BIN
            , [PurchaseOrder] Varchar(35)    Collate Latin1_General_BIN
            , [Line] Int
            , [MStockCode] Varchar(35)        Collate Latin1_General_BIN
            , [MStockDes] Varchar(150)        Collate Latin1_General_BIN
            , [MOrderQty] Numeric(20 , 8)
            , [MPrice] Numeric(20 , 3)
            );

--create script to pull data from each db into the tables

        Declare @SQL1 Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end'
            + --Only query DBs beginning SysProCompany

            '
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN'
            + --only companies selected in main run, or if companies selected then
all

            '
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT'
            + --count number of tables requested (number of commas plus one)

            '
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')'
            + --Count of the tables requested how many exist in the db

            '
```

Author: Johnson, Chris

```
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'','')) '
            + --only if the count matches (all the tables exist in the requested db)
then run the script

            '
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
                Insert #PorMasterHdr
                        ( DatabaseName
                        , PurchaseOrder
                        , OrderEntryDate
                        , OrderDueDate
                        , OrderStatus
                        , Supplier
                        , CancelledFlag
                        )
                SELECT DatabaseName = @DBCode
                     , PurchaseOrder
                     , OrderEntryDate
                     , OrderDueDate
                     , OrderStatus
                     , Supplier
                     , CancelledFlag FROM PorMasterHdr
            End
    End';
        Declare @SQL2 Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end'
            + --Only query DBs beginning SysProCompany

            '
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN'
            + --only companies selected in main run, or if companies selected then
all

            '
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT'
            + --count number of tables requested (number of commas plus one)

            '
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')'
            + --Count of the tables requested how many exist in the db

            '
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'','')) '
            + --only if the count matches (all the tables exist in the requested db)
then run the script
```

Author: Johnson, Chris

```
            '
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
                    Insert #ApSupplier
                            ( DatabaseName
                            , Supplier
                            , SupplierName
                            )
                    SELECT DatabaseName = @DBCode
                         , Supplier
                         , SupplierName
                         FROM ApSupplier
                            End
    End';
        Declare @SQL3 Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end'
            + --Only query DBs beginning SysProCompany

            '
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN'
            + --only companies selected in main run, or if companies selected then
all

            '
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT'
            + --count number of tables requested (number of commas plus one)

            '
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')'
            + --Count of the tables requested how many exist in the db

            '
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'','')) '
            + --only if the count matches (all the tables exist in the requested db)
then run the script

            '
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
                Insert #PorMasterDetail
                        ( DatabaseName
                        , PurchaseOrder
                        , Line
                        , MStockCode
                        , MStockDes
                        , MOrderQty
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-OutstandingPurchaseOrders

```
                        , MPrice
                        )
                SELECT DatabaseName = @DBCode
                     , PurchaseOrder
                     , Line
                     , MStockCode
                     , MStockDes
                     , MOrderQty
                     , MPrice FROM  PorMasterDetail
                        End
    End';

--Enable this function to check script changes (try to run script directly against
db manually)

--Print @SQL1


--execute script against each db, populating the base tables

        Exec [Process].[ExecForEachDB] @cmd = @SQL1;
        Exec [Process].[ExecForEachDB] @cmd = @SQL2;
        Exec [Process].[ExecForEachDB] @cmd = @SQL3;

--define the results you want to return

        Create Table [#Results]
            (
              [DatabaseName] Varchar(150)            collate Latin1_General_BIN
            , [PurchaseOrder] Varchar(35)            collate Latin1_General_BIN
            , [SupplierName] Varchar(150)            collate Latin1_General_BIN
            , [OrderStatusDescription] Varchar(150)    collate Latin1_General_BIN
            , [OrderEntryDate] DateTime2
            , [OrderDueDate] DateTime2
            , [Line] Int
            , [StockCode] Varchar(35)                collate Latin1_General_BIN
            , [StockDesc] Varchar(150)                collate Latin1_General_BIN
            , [OrderQty] Numeric(20 , 6)
            , [Price] Numeric(20 , 3)
            );

--Placeholder to create indexes as required


--script to combine base data and insert into results table

        Insert  [#Results]
                ( [DatabaseName]
                , [PurchaseOrder]
                , [SupplierName]
                , [OrderStatusDescription]
                , [OrderEntryDate]
                , [OrderDueDate]
                , [Line]
                , [StockCode]
                , [StockDesc]
                , [OrderQty]
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-OutstandingPurchaseOrders

```
                , [Price]
                )
                Select  [PH].[DatabaseName]
                      , [PH].[PurchaseOrder]
                      , [APS].[SupplierName]
                      , [PS].[OrderStatusDescription]
                      , [PH].[OrderEntryDate]
                      , [PH].[OrderDueDate]
                      , [PMD].[Line]
                      , [PMD].[MStockCode]
                      , [PMD].[MStockDes]
                      , [PMD].[MOrderQty]
                      , [PMD].[MPrice]
                From    [#PorMasterHdr] [PH]
                        Left Join [BlackBox].[Lookups].[PurchaseOrderStatus] [PS] On
[PH].[OrderStatus] = [PS].[OrderStatusCode]
                                                                And [PS].[Company] =
[PH].[DatabaseName]
                        Left Join [BlackBox].[Lookups].[PurchaseOrderInvoiceMapping]
[POI] On [POI].[PurchaseOrder] = [PH].[PurchaseOrder]
                                                                And [POI].[Company] =
[PH].[DatabaseName]
                        Left Join [#ApSupplier] [APS] On [APS].[Supplier] =
[PH].[Supplier]
                                                            And [APS].[DatabaseName] =
[PH].[DatabaseName]
                        Left Join [#PorMasterDetail] [PMD] On [PMD].[PurchaseOrder]
= [PH].[PurchaseOrder]
                Where   [POI].[Invoice] Is Null --Where an invoice has not been
received

                        And [PH].[CancelledFlag] <> 'Y' --Ignore cancelled flag

        ;

--return results

        Select  [DatabaseName]
              , [PurchaseOrder]
              , [SupplierName]
              , [OrderStatusDescription]
              , [OrderEntryDate] = Cast([OrderEntryDate] As Date)
              , [OrderDueDate] = Cast([OrderDueDate] As Date)
              , [Line] = Coalesce([Line] , 0)
              , [StockCode]
              , [StockDesc]
              , [OrderQty] = Coalesce([OrderQty] , 0)
              , [Price] = Coalesce([Price] , 0)
              , [Status] = Case When [OrderDueDate] Is Null
                                Then 'No due date specified'
                                When [OrderDueDate] <= GetDate()
                                Then 'Overdue - nothing received'
                                Else 'Not overdue - nothing received'
                           End
        From    [#Results]
        Where   Coalesce([StockCode] , '') <> ''
                And Coalesce([StockDesc] , '') <> ''
                And Coalesce([OrderQty] , 0) <> 0
                And Coalesce([Price] , 0) <> 0
```

Author: Johnson, Chris

```
        Order By [OrderDueDate] Desc
              , [OrderEntryDate] Desc;


    End;


GO
EXEC sp_addextendedproperty N'MS_Description', N'list of purchase orders
outstanding', 'SCHEMA', N'Report', 'PROCEDURE', N'UspResults_OutstandingPurchase-
Orders', NULL, NULL
GO
```

Uses

[Lookups].[PurchaseOrderInvoiceMapping]
[Lookups].[PurchaseOrderStatus]
[Process].[ExecForEachDB]
[Process].[UspInsert_RedTagLogs]
[Report]

[Report].[UspResults_PaymentRunVerify]

MS_Description

data for Payment Run report

Parameters

Name	Data Type	Max Length (Bytes)
@Company	varchar(max)	max
@RedTagType	char	1
@RedTagUse	varchar(500)	500

SQL Script

```
CREATE Proc [Report].[UspResults_PaymentRunVerify]
    (
      @Company Varchar(Max)
    , @RedTagType Char(1)
    , @RedTagUse Varchar(500)
    )
As
    Begin
/*
Template designed by Chris Johnson, Prometic Group September 2015
Stored procedure for Payment run verify
--Exec [Report].[UspResults_PaymentRunVerify]  10
*/
        If IsNumeric(@Company) = 0
            Begin
                Select  @Company = Upper(@Company);
            End;
--remove nocount on to speed up query

        Set NoCount On;

--Red tag

        Declare @RedTagDB Varchar(255)= Db_Name();
        Exec [Process].[UspInsert_RedTagLogs] @StoredProcDb = 'BlackBox' ,
            @StoredProcSchema = 'Report' ,
            @StoredProcName = 'UspResults_PaymentRunVerify' ,
            @UsedByType = @RedTagType , @UsedByName = @RedTagUse ,
            @UsedByDb = @RedTagDB;

--list the tables that are to be pulled back from each DB - if they are not found
the script will not be run against that db

        Declare @ListOfTables Varchar(Max) = 'ApInvoice,ApJnlSummary,ApJnlDistrib,Ap-
PayRunDet,ApPayRunHdr';

--create temporary tables to be pulled from different databases, including a column
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-PaymentRunVerify

```
to id

        Create Table [#ApInvoice]
            (
              [DatabaseName] Varchar(150) Collate Latin1_General_BIN
            , [Supplier] Varchar(15) Collate Latin1_General_BIN
            , [Invoice] Varchar(20) Collate Latin1_General_BIN
            , [PostCurrency] Char(3) Collate Latin1_General_BIN
            , [ConvRate] Float
            , [MulDiv] Char(1) Collate Latin1_General_BIN
            , [MthInvBal1] Float
            , [InvoiceYear] Int
            , [InvoiceMonth] Int
            , [JournalDate] DateTime2
            , [InvoiceDate] DateTime2
            );
        Create Table [#ApJnlSummary]
            (
              [DatabaseName] Varchar(150) Collate Latin1_General_BIN
            , [Supplier] Varchar(15) Collate Latin1_General_BIN
            , [Invoice] Varchar(20) Collate Latin1_General_BIN
            , [TrnYear] Int
            , [TrnMonth] Int
            , [Journal] Int
            , [EntryNumber] Int
            );
        Create Table [#ApJnlDistrib]
            (
              [DatabaseName] Varchar(150) Collate Latin1_General_BIN
            , [DistrValue] Float
            , [ExpenseGlCode] Varchar(35) Collate Latin1_General_BIN
            , [TrnYear] Int
            , [TrnMonth] Int
            , [Journal] Int
            , [EntryNumber] Int
            );
        Create Table [#ApPayRunDet]
            (
              [DatabaseName] Varchar(150) Collate Latin1_General_BIN
            , [Supplier] Varchar(15)     Collate Latin1_General_BIN
            , [Invoice] Varchar(20)        Collate Latin1_General_BIN
            , [Cheque] Varchar(15)         Collate Latin1_General_BIN
            , [ChequeDate] DateTime2
            , [InvoiceDate] DateTime2
            , [NetPayValue] Float
            , [DueDate] DateTime2
            , [InvoiceType] Char(1)     Collate Latin1_General_BIN
            , [PostValue] Float
            , [PostCurrency] Char(3) Collate Latin1_General_BIN
            , [PostConvRate] Float
            , [PostMulDiv] Char(1) Collate Latin1_General_BIN
            , [SupplierName] Varchar(50) Collate Latin1_General_BIN
            , [PaymentNumber] Varchar(15) Collate Latin1_General_BIN
            );
        Create Table [#ApPayRunHdr]
            (
              [DatabaseName] Varchar(150)  Collate Latin1_General_BIN
```

Author: Johnson, Chris

```
            , [PaymentNumber] Varchar(15)  Collate Latin1_General_BIN
            , [Bank] Varchar(15) Collate Latin1_General_BIN
            , [PaymentDate] DateTime2
            , [PayYear] Int
            , [PayMonth] Int
            , [Operator] Varchar(20) Collate Latin1_General_BIN
            , [ChRegister] Float
            );
        Create Table [#ApSupplier]
            (
              [DatabaseName] Varchar(150) Collate Latin1_General_BIN
            , [Supplier] Varchar(50) Collate Latin1_General_BIN
            , [SupplierName] Varchar(255) Collate Latin1_General_BIN
            , [Currency] Varchar(5) Collate Latin1_General_BIN
            );

--create script to pull data from each db into the tables

        Declare @SQL1 Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end'
            + --Only query DBs beginning SysProCompany

            '
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN'
            + --only companies selected in main run, or if companies selected then
all

            '
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT'
            + --count number of tables requested (number of commas plus one)

            '
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')'
            + --Count of the tables requested how many exist in the db

            '
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'','')) '
            + --only if the count matches (all the tables exist in the requested db)
then run the script

            '
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
                    Insert #ApInvoice
                                ( DatabaseName
                                , Supplier
                                , Invoice
```

```
                                , PostCurrency
                                , ConvRate
                                , MulDiv
                                , MthInvBal1
                                , InvoiceYear
                                , InvoiceMonth
                                , JournalDate
                                , InvoiceDate
                                )
                        SELECT @DBCode
                                  , Supplier
                                  , Invoice
                                  , PostCurrency
                                  , ConvRate
                                  , MulDiv
                                  , MthInvBal1
                                  , InvoiceYear
                                  , InvoiceMonth
                                  , JournalDate
                                  , InvoiceDate     FROM ApInvoice

                        Insert #ApJnlSummary
                                ( DatabaseName
                                , Supplier
                                , Invoice
                                , TrnYear
                                , TrnMonth
                                , Journal
                                , EntryNumber
                                )
                        SELECT @DBCode
                                ,Supplier
                                ,Invoice
                                ,TrnYear
                                ,TrnMonth
                                ,Journal
                                ,EntryNumber
                         FROM ApJnlSummary
                         Where [TransactionCode] not in (''X'')
                End
    End';
        Declare @SQL2 Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end'
            + --Only query DBs beginning SysProCompany

            '
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN'
            + --only companies selected in main run, or if companies selected then
all

            '
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
```

Author: Johnson, Chris

```
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT'
            + --count number of tables requested (number of commas plus one)

            '
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')'
            + --Count of the tables requested how many exist in the db

            '
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'','')) '
            + --only if the count matches (all the tables exist in the requested db)
then run the script

            '
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
                        Insert #ApJnlDistrib
                                ( DatabaseName
                                , DistrValue
                                , ExpenseGlCode
                                , TrnYear
                                , TrnMonth
                                , Journal
                                , EntryNumber
                                )
                        SELECT @DBCode
                                ,DistrValue
                                ,ExpenseGlCode
                                ,TrnYear
                                ,TrnMonth
                                ,Journal
                                ,EntryNumber
                         FROM ApJnlDistrib

                         Insert #ApPayRunDet
                                ( DatabaseName
                                , Supplier
                                , Invoice
                                , Cheque
                                , ChequeDate
                                , InvoiceDate
                                , NetPayValue
                                , DueDate
                                , InvoiceType
                                , PostValue
                                , PostCurrency
                                , PostConvRate
                                , PostMulDiv
                                , SupplierName
                                , PaymentNumber
                                )
                        SELECT @DBCode
                                ,Supplier
                                ,Invoice
```

Author: Johnson, Chris

```
                                ,Cheque
                                ,ChequeDate
                                ,InvoiceDate
                                ,NetPayValue
                                ,DueDate
                                ,InvoiceType
                                ,PostValue
                                ,PostCurrency
                                ,PostConvRate
                                ,PostMulDiv
                                ,SupplierName
                                ,PaymentNumber
                        FROM ApPayRunDet
            End
    End';
        Declare @SQL3 Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end'
            + --Only query DBs beginning SysProCompany

            '
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN'
            + --only companies selected in main run, or if companies selected then
all

            '
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT'
            + --count number of tables requested (number of commas plus one)

            '
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')'
            + --Count of the tables requested how many exist in the db

            '
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'','')) '
            + --only if the count matches (all the tables exist in the requested db)
then run the script

            '
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
                Insert #ApPayRunHdr
                            ( DatabaseName
                            , PaymentNumber
                            , Bank
                            , PaymentDate
                            , PayYear
                            , PayMonth
```

Author: Johnson, Chris

```
                            , Operator
                            , ChRegister
                            )
                SELECT @DBCode
                            , PaymentNumber
                            , Bank
                            , PaymentDate
                            , PayYear
                            , PayMonth
                            , Operator
                            , ChRegister
                FROM ApPayRunHdr
            End
    End';
        Declare @SQLApSupplier Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end'
            + --Only query DBs beginning SysProCompany

            '
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN'
            + --only companies selected in main run, or if companies selected then
all

            '
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT'
            + --count number of tables requested (number of commas plus one)

            '
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')'
            + --Count of the tables requested how many exist in the db

            '
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'','')) '
            + --only if the count matches (all the tables exist in the requested db)
then run the script

            '
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
                Insert [#ApSupplier]
                        ( [DatabaseName]
                        , [Supplier]
                        , [SupplierName]
                        , [Currency]
                        )
                SELECT [DatabaseName]=@DBCode
                     , [AS].[Supplier]
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-PaymentRunVerify

```
                    , [AS].[SupplierName]
                    , [AS].[Currency]
                     FROM [ApSupplier] As [AS]
            End
    End';
--Enable this function to check script changes (try to run script directly against
db manually)

--Print @SQL


--execute script against each db, populating the base tables

        Exec [Process].[ExecForEachDB] @cmd = @SQL1;
        Exec [Process].[ExecForEachDB] @cmd = @SQL2;
        Exec [Process].[ExecForEachDB] @cmd = @SQL3;
        Exec [Process].[ExecForEachDB] @cmd = @SQLApSupplier;

--define the results you want to return

        Create Table [#Results]
            (
              [DatabaseName] Varchar(150)         Collate Latin1_General_BIN
            , [Supplier] Varchar(15)             Collate Latin1_General_BIN
            , [Invoice] Varchar(20)                Collate Latin1_General_BIN
            , [PostCurrency] Char(3)             Collate Latin1_General_BIN
            , [ConvRate] Float
            , [MulDiv] Char(1)                     Collate Latin1_General_BIN
            , [MthInvBal] Float
            , [CompLocalAmt] Decimal(10 , 2)
            , [DistrValue] Float
            , [Description] Varchar(50)              Collate Latin1_General_BIN
            , [ExpenseGlCode] Varchar(35)         Collate Latin1_General_BIN
            , [InvoiceYear] Int
            , [InvoiceMonth] Int
            , [JournalDate] DateTime2
            , [InvoiceDate] DateTime2
            , [PaymentNumber] Int
            , [Bank] Varchar(15)                 Collate Latin1_General_BIN
            , [PaymentDate] DateTime2
            , [PayYear] Int
            , [PayMonth] Int
            , [Operator] Varchar(20)             Collate Latin1_General_BIN
            , [ChRegister] Float
            , [Cheque] Varchar(15)                 Collate Latin1_General_BIN
            , [ChequeDate] DateTime2
            --, InvoiceDate DATETIME2

            , [InvNetPayValue] Decimal(15 , 3)
            , [DueDate] DateTime2
            , [InvoiceType] Char(1)                  Collate Latin1_General_BIN
            , [PostValue] Decimal(15 , 3)
            , [PostConvRate] Float
            , [PostMulDiv] Char(1)                 Collate Latin1_General_BIN
            , [SupplierName] Varchar(50)         Collate Latin1_General_BIN
            , [SupplierCurrency] Varchar(5)          Collate Latin1_General_BIN
            );
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-PaymentRunVerify

```
--Placeholder to create indexes as required


--script to combine base data and insert into results table

        Insert  [#Results]
                ( [DatabaseName]
                , [Supplier]
                , [Invoice]
                , [PostCurrency]
                , [ConvRate]
                , [MulDiv]
                , [MthInvBal]
                , [CompLocalAmt]
                , [DistrValue]
                , [Description]
                , [ExpenseGlCode]
                , [InvoiceYear]
                , [InvoiceMonth]
                , [JournalDate]
                , [InvoiceDate]
                , [PaymentNumber]
                , [Bank]
                , [PaymentDate]
                , [PayYear]
                , [PayMonth]
                , [Operator]
                , [ChRegister]
                , [Cheque]
                , [ChequeDate]
                , [InvNetPayValue]
                , [DueDate]
                , [InvoiceType]
                , [PostValue]
                , [PostConvRate]
                , [PostMulDiv]
                , [SupplierName]
                , [SupplierCurrency]
                )
               Select  [AI].[DatabaseName]
                     , [AI].[Supplier]
                     , [AI].[Invoice]
                     , [AI].[PostCurrency]
                     , [AI].[ConvRate]
                     , [AI].[MulDiv]
                     , [MthInvBal] = [AI].[MthInvBal1]
                     , [CompLocalAmt] = Cast(Case When [AI].[MulDiv] = 'M'
                                                  Then [AI].[MthInvBal1]
                                                       * [AI].[ConvRate]
                                                  Else [AI].[MthInvBal1]
                                                       / [AI].[ConvRate]
                                             End As Decimal(10 , 2))
                     , [DistrValue] = Sum([AJD].[DistrValue])
                     , [GM].[Description]
                     , [ExpenseGlCode] = Case When [AJD].[ExpenseGlCode] = ''
                                              Then Null
```

Author: Johnson, Chris

```
                                            Else [AJD].[ExpenseGlCode]
                                       End
                  , [AI].[InvoiceYear]
                  , [AI].[InvoiceMonth]
                  , [AI].[JournalDate]
                  , [AI].[InvoiceDate]
                  , [APH].[PaymentNumber]
                  , [APH].[Bank]
                  , [APH].[PaymentDate]
                  , [APH].[PayYear]
                  , [APH].[PayMonth]
                  , [APH].[Operator]
                  , [APH].[ChRegister]
                  , [APD].[Cheque]
                  , [APD].[ChequeDate]
                  , [InvNetPayValue] = [APD].[NetPayValue]
                  , [APD].[DueDate]
                  , [APD].[InvoiceType]
                  , [APD].[PostValue]
                  , [APD].[PostConvRate]
                  , [APD].[PostMulDiv]
                  , [APD].[SupplierName]
                  , [SupplierCurrency] = [AS].[Currency]
            From    [#ApInvoice] [AI]
                    Left Join [#ApJnlSummary] [AJS] With ( NoLock ) On
[AI].[Supplier] = [AJS].[Supplier]
                                                       And [AI].[Invoice] =
[AJS].[Invoice]
                                                       And [AJS].[Database-
Name] = [AI].[DatabaseName]
                    Left Join [#ApJnlDistrib] [AJD] With ( NoLock ) On
[AJD].[TrnYear] = [AJS].[TrnYear]
                                                       And [AJD].[TrnMonth] =
[AJS].[TrnMonth]
                                                       And [AJD].[Journal] =
[AJS].[Journal]
                                                       And [AJD].[Entry-
Number] = [AJS].[EntryNumber]
                                                       And [AJD].[Database-
Name] = [AJS].[DatabaseName]
                    Left Join [SysproCompany40].[dbo].[GenMaster] [GM] On
[GM].[GlCode] = [AJD].[ExpenseGlCode] Collate Latin1_General_BIN
                    Inner Join [#ApPayRunDet] [APD] On [APD].[Supplier] =
[AI].[Supplier]
                                                         And [APD].[Invoice] =
[AI].[Invoice]
                                                         And [APD].[DatabaseName]
= [AI].[DatabaseName]
                    Left Join [#ApPayRunHdr] [APH] On [APH].[PaymentNumber] =
[APD].[PaymentNumber]
                                                        And [APH].[PaymentNumber]
= [APD].[PaymentNumber]
                    Left Join [#ApSupplier] As [AS] On [AS].[Supplier] =
[AI].[Supplier]
                                                         And [AS].[DatabaseName] =
[AI].[DatabaseName]
            Group By [AI].[DatabaseName]
                  , [AJD].[TrnYear]
                  , [AJD].[TrnMonth]
                  , Case When [AJD].[ExpenseGlCode] = '' Then Null
                         Else [AJD].[ExpenseGlCode]
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-PaymentRunVerify

```
                        End
                      , [GM].[Description]
                      , [AJS].[Supplier]
                      , [AJS].[Invoice]
                      , [AI].[Supplier]
                      , [AI].[Invoice]
                      , [AI].[PostCurrency]
                      , [AI].[ConvRate]
                      , [AI].[MulDiv]
                      , [AI].[MthInvBal1]
                      , Cast(Case When [AI].[MulDiv] = 'M'
                                  Then [AI].[MthInvBal1] * [AI].[ConvRate]
                                  Else [AI].[MthInvBal1] / [AI].[ConvRate]
                             End As Decimal(10 , 2))
                      , [AI].[InvoiceYear]
                      , [AI].[InvoiceMonth]
                      , [AI].[JournalDate]
                      , [AI].[InvoiceDate]
                      , [APH].[PaymentNumber]
                      , [APH].[Bank]
                      , [APH].[PaymentDate]
                      , [APH].[PayYear]
                      , [APH].[PayMonth]
                      , [APH].[Operator]
                      , [APH].[ChRegister]
                      , [APD].[Cheque]
                      , [APD].[ChequeDate]
                      , [APD].[InvoiceDate]
                      , [APD].[NetPayValue]
                      , [APD].[DueDate]
                      , [APD].[InvoiceType]
                      , [APD].[PostValue]
                      , [APD].[PostCurrency]
                      , [APD].[PostConvRate]
                      , [APD].[PostMulDiv]
                      , [APD].[SupplierName]
                      , [AS].[Currency];

--return results

        Select  [Company] = [R].[DatabaseName]
              , [R].[Supplier]
              , [R].[Invoice]
              , [R].[PostCurrency]
              , [R].[ConvRate]
              , [R].[MulDiv]
              , [R].[MthInvBal]
              , [R].[CompLocalAmt]
              , [R].[DistrValue]
              , [R].[Description]
              , [R].[ExpenseGlCode]
              , [R].[InvoiceYear]
              , [R].[InvoiceMonth]
              , [JournalDate] = Convert(Date , [R].[JournalDate])
              , [R].[PaymentNumber]
              , [R].[Bank]
              , [PaymentDate] = Convert(Date , [R].[PaymentDate])
```

Author: Johnson, Chris

```
                , [R].[PayYear]
                , [R].[PayMonth]
                , [R].[Operator]
                , [R].[ChRegister]
                , [R].[Cheque]
                , [ChequeDate] = Convert(Date , [R].[ChequeDate])
                , [InvoiceDate] = Convert(Date , [R].[InvoiceDate])
                , [R].[InvNetPayValue]
                , [DueDate] = Convert(Date , [R].[DueDate])
                , [R].[InvoiceType]
                , [R].[PostValue]
                , [R].[PostConvRate]
                , [R].[PostMulDiv]
                , [R].[SupplierName]
                , [ExpenseType] = Null
                , [cn].[CompanyName]
                , [R].[SupplierCurrency]
        From    [#Results] [R]
                Left Join [Lookups].[CompanyNames] As [cn] On [cn].[Company] =
[R].[DatabaseName] Collate Latin1_General_BIN;

    End;


GO
EXEC sp_addextendedproperty N'MS_Description', N'data for Payment Run report',
'SCHEMA', N'Report', 'PROCEDURE', N'UspResults_PaymentRunVerify', NULL, NULL
GO
```

Uses

[Lookups].[CompanyNames]
[Process].[ExecForEachDB]
[Process].[UspInsert_RedTagLogs]
[Report]

Author: Johnson, Chris

[Report].[UspResults_PosNoReqs]

MS_Description

details of Purchase orders without requisitions

Parameters

Name	Data Type	Max Length (Bytes)
@Company	varchar(max)	max
@RedTagType	char	1
@RedTagUse	varchar(500)	500

SQL Script

```
CREATE Proc [Report].[UspResults_PosNoReqs]
    (
      @Company Varchar(Max)
    , @RedTagType Char(1)
    , @RedTagUse Varchar(500)
    )
As
    Begin
        Set NoCount On;

        If IsNumeric(@Company) = 0
            Begin
                Select  @Company = Upper(@Company);
            End;

--Red tag

        Declare @RedTagDB Varchar(255)= Db_Name();
        Exec [Process].[UspInsert_RedTagLogs] @StoredProcDb = 'BlackBox' ,
            @StoredProcSchema = 'Report' ,
            @StoredProcName = 'UspResults_PosNoReqs' ,
            @UsedByType = @RedTagType , @UsedByName = @RedTagUse ,
            @UsedByDb = @RedTagDB;

--list the tables that are to be pulled back from each DB - if they are not found
the script will not be run against that db

        Declare @ListOfTables Varchar(Max) = 'PorMasterHdr,PorMasterDetail';

--create temporary tables to be pulled from different databases, including a column
to id

        Create Table [#PorMasterDetail]
            (
              [DatabaseName] Varchar(150)        collate latin1_general_bin
            , [PurchaseOrder] Varchar(20)        collate latin1_general_bin
            , [MRequisition] Varchar(10)        collate latin1_general_bin
```

Author: Johnson, Chris

```
            );
        Create Table [#PorMasterHdr]
            (
              [DatabaseName] Varchar(150)        collate latin1_general_bin
            , [PurchaseOrder] Varchar(20)        collate latin1_general_bin
            , [OrderStatus] Char(1)                collate latin1_general_bin
            , [Supplier] Varchar(15)            collate latin1_general_bin
            , [OrderEntryDate] Date
            , [OrderDueDate] Date
            , [DatePoCompleted] Date
            , [Buyer] Varchar(20)                collate latin1_general_bin
            , [CancelledFlag] Char(1)            collate latin1_general_bin
            );




--create script to pull data from each db into the tables

        Declare @SQLPorMasterDetail Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            BEGIN
                Insert [#PorMasterDetail]
                        ( [DatabaseName]
                        , [PurchaseOrder]
                        , [MRequisition]
                        )
                SELECT [DatabaseName]=@DBCode
                     , [PMD].[PurchaseOrder]
                     , [PMD].[MRequisition] FROM [PorMasterDetail] [PMD]
            End
    End';
        Declare @SQLPorMasterHdr Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            BEGIN
                Insert [#PorMasterHdr]
                        ( [DatabaseName]
                        , [PurchaseOrder]
                        , [OrderStatus]
                        , [Supplier]
                        , [OrderEntryDate]
                        , [OrderDueDate]
                        , [DatePoCompleted]
```

```
                        , [Buyer]
                        , [CancelledFlag]
                        )
                SELECT [DatabaseName]=@DBCode
                     , [PMH].[PurchaseOrder]
                     , [PMH].[OrderStatus]
                     , [PMH].[Supplier]
                     , [PMH].[OrderEntryDate]
                     , [PMH].[OrderDueDate]
                     , [PMH].[DatePoCompleted]
                     , [PMH].[Buyer]
                     , [PMH].[CancelledFlag]
                FROM [PorMasterHdr] [PMH]
            End
    End';

--execute script against each db, populating the base tables

        Exec [Process].[ExecForEachDB_WithTableCheck] @cmd = @SQLPorMasterDetail ,
            @SchemaTablesToCheck = @ListOfTables;
        Exec [Process].[ExecForEachDB_WithTableCheck] @cmd = @SQLPorMasterHdr ,
            @SchemaTablesToCheck = @ListOfTables;

--define the results you want to return

        Create Table [#Results]
            (
              [DatabaseName] Varchar(150)        collate latin1_general_bin
            , [PurchaseOrder] Varchar(20)        collate latin1_general_bin
            , [OrderStatus] Varchar(150)        collate latin1_general_bin
            , [Supplier] Varchar(15)            collate latin1_general_bin
            , [OrderEntryDate] Date
            , [OrderDueDate] Date
            , [DatePoCompleted] Date
            , [Buyer] Varchar(20)                collate latin1_general_bin
            );

--Placeholder to create indexes as required


--script to combine base data and insert into results table

        Insert  [#Results]
                ( [DatabaseName]
                , [PurchaseOrder]
                , [OrderStatus]
                , [Supplier]
                , [OrderEntryDate]
                , [OrderDueDate]
                , [DatePoCompleted]
                , [Buyer]
                )
                Select  [PMH].[DatabaseName]
                      , [PMH].[PurchaseOrder]
                      , [OrderStatus] = [POS].[OrderStatusDescription]
                      , [PMH].[Supplier]
                      , [PMH].[OrderEntryDate]
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_Pos-NoReqs

```
                      , [PMH].[OrderDueDate]
                      , [PMH].[DatePoCompleted]
                      , [PMH].[Buyer]
                From    [#PorMasterDetail] [PMD]
                        Inner Join [#PorMasterHdr] [PMH]
                            On [PMH].[PurchaseOrder] = [PMD].[PurchaseOrder]
                               And [PMH].[DatabaseName] = [PMD].[DatabaseName]
                        Left Join [BlackBox].[Lookups].[PurchaseOrderStatus] [POS]
                            On [PMH].[OrderStatus] = [POS].[OrderStatusCode]
                               And [POS].[Company] = [PMH].[DatabaseName]
                Where   Coalesce([PMH].[CancelledFlag] , 'N') <> 'Y'
                Group By [PMH].[DatabaseName]
                      , [PMH].[PurchaseOrder]
                      , [POS].[OrderStatusDescription]
                      , [PMH].[Supplier]
                      , [PMH].[OrderEntryDate]
                      , [PMH].[OrderDueDate]
                      , [PMH].[DatePoCompleted]
                      , [PMH].[Buyer]
                Having  Count(Distinct Case When Coalesce([PMD].[MRequisition] ,
                                                          '') = '' Then Null
                                            Else [PMD].[MRequisition]
                                       End) = 0;

        Set NoCount Off;
--return results

        Select  [R].[DatabaseName]
              , [CN].[CompanyName]
              , [CN].[ShortName]
              , [PurchaseOrder] = Case When IsNumeric([R].[PurchaseOrder]) = 1
                                       Then Convert(Varchar(20) , Convert(BigInt ,
[R].[PurchaseOrder]))
                                       Else [R].[PurchaseOrder]
                                  End
              , [R].[OrderStatus]
              , [R].[Supplier]
              , [R].[OrderEntryDate]
              , [R].[OrderDueDate]
              , [R].[DatePoCompleted]
              , [Buyer] = Case When [R].[Buyer] = '' Then Null
                               Else [R].[Buyer]
                          End
        From    [#Results] [R]
                Left Join [Lookups].[CompanyNames] [CN]
                    On [R].[DatabaseName] = [CN].[Company];

    End;

GO
EXEC sp_addextendedproperty N'MS_Description', N'details of Purchase orders without
requisitions', 'SCHEMA', N'Report', 'PROCEDURE', N'UspResults_PosNoReqs', NULL, NULL
GO
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_Pos-NoReqs

Uses

[Lookups].[CompanyNames]
[Lookups].[PurchaseOrderStatus]
[Process].[ExecForEachDB_WithTableCheck]
[Process].[UspInsert_RedTagLogs]
[Report]

Author: Johnson, Chris

[Report].[UspResults_PurchaseOrderChanges]

MS_Description

list of changes made to purchase orders

Parameters

Name	Data Type	Max Length (Bytes)
@Company	varchar(max)	max
@RedTagType	char	1
@RedTagUse	varchar(500)	500

SQL Script

```
CREATE Proc [Report].[UspResults_PurchaseOrderChanges]
    (
      @Company Varchar(Max)
    , @RedTagType Char(1)
    , @RedTagUse Varchar(500)
    )
As
    Begin
/*
Template designed by Chris Johnson, Prometic Group September 2015
Stored procedure set out to query multiple databases with the same information and
return it in a collated format
--Exec  [Report].[UspResults_PurchaseOrderChanges] 10
*/
        Set NoCount Off;
        If IsNumeric(@Company) = 0
            Begin
                Select  @Company = Upper(@Company);
            End;

--remove nocount on to speed up query

        Set NoCount On;
--Red tag

        Declare @RedTagDB Varchar(255)= Db_Name();
        Exec [Process].[UspInsert_RedTagLogs] @StoredProcDb = 'BlackBox' ,
            @StoredProcSchema = 'Report' ,
            @StoredProcName = 'UspResults_PurchaseOrderChanges' ,
            @UsedByType = @RedTagType , @UsedByName = @RedTagUse ,
            @UsedByDb = @RedTagDB;
--grab and unpivot all audit tables in BlackBox History Tables


--list the tables that are to be pulled back from each DB - if they are not found
the script will not be run against that db
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-PurchaseOrderChanges

```
        Declare @ListOfTables Varchar(Max) = 'PorMasterHdr';

--create temporary tables to be pulled from different databases, including a column to id

        Create Table [#PorMasterHdr]
            (
              [DatabaseName] Varchar(150)        collate latin1_general_bin
            , [PurchaseOrder] Varchar(35)        collate latin1_general_bin
            , [OrderEntryDate] DateTime2
            , [OrderDueDate] DateTime2
            );

--create script to pull data from each db into the tables

        Declare @SQL Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-Name(),len(db_Name())-13) else null end
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN'
            + --only companies selected in main run, or if companies selected then all

            '
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT'
            + --count number of tables requested (number of commas plus one)

            '
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-String](@ListOfTables,'','')'
            + --Count of the tables requested how many exist in the db

            '
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-Box.dbo.udf_SplitString(@ListOfTables,'','')) '
            + --only if the count matches (all the tables exist in the requested db) then run the script

            '
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
                    Insert #PorMasterHdr
                            ( DatabaseName
                            , PurchaseOrder
                            , OrderEntryDate
                            , OrderDueDate
                            )
                    SELECT DatabaseName = @DB
                         , PurchaseOrder
                         , OrderEntryDate
                         , OrderDueDate FROM PorMasterHdr
```

Author: Johnson, Chris

```
            End
    End';

--Enable this function to check script changes (try to run script directly against
db manually)



--execute script against each db, populating the base tables

        Exec [Process].[ExecForEachDB] @cmd = @SQL;

--define the results you want to return

        Create Table [#Results]
            (
              [ItemKey] Varchar(150)                    collate latin1_general_bin
            , [PurchaseOrder] Varchar(35)                collate latin1_general_bin
            , [Line] Varchar(35)                        collate latin1_general_bin
            , [OrderEntryDate] DateTime2
            , [OrderDueDate] DateTime2
            , [DatabaseName] Varchar(150)                collate latin1_general_bin
            , [TransactionDescription] Varchar(200)        collate
latin1_general_bin
            , [SignatureDatetime] DateTime2
            , [Operator] Varchar(50)                    collate latin1_general_bin
            , [PreviousPrice] Float
            , [Price] Float
            , [PreviousForeignPrice] Float
            , [ForeignPrice] Float
            , [PreviousQuantity] Float
            , [Quantity] Float
            , [OrderUnitOfMeasure] Varchar(15)            collate latin1_general_bin
            , [PriceDiff] Float
            , [PriceDiffPercent] Float
            , [ForeignPriceDiff] Float
            , [ForeignPriceDiffPercent] Float
            , [QuantityDiff] Float
            , [QuantityDiffPercent] Float
            , [LineForeignValue] Float
            , [PrevLineForeignValue] Float
            , [LineLocalValue] Float
            , [PrevLineLocalValue] Float
            , [CompanyName] Varchar(255)                collate latin1_general_bin
            );

--Placeholder to create indexes as required



--script to combine base data and insert into results table

        Insert  [#Results]
                ( [ItemKey]
                , [PurchaseOrder]
                , [Line]
                , [OrderEntryDate]
                , [OrderDueDate]
```

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-PurchaseOrderChanges

```
                , [DatabaseName]
                , [TransactionDescription]
                , [SignatureDatetime]
                , [Operator]
                , [PreviousPrice]
                , [Price]
                , [PreviousForeignPrice]
                , [ForeignPrice]
                , [PreviousQuantity]
                , [Quantity]
                , [OrderUnitOfMeasure]
                , [PriceDiff]
                , [PriceDiffPercent]
                , [ForeignPriceDiff]
                , [ForeignPriceDiffPercent]
                , [QuantityDiff]
                , [QuantityDiffPercent]
                , [LineForeignValue]
                , [PrevLineForeignValue]
                , [LineLocalValue]
                , [PrevLineLocalValue]
                , [CompanyName]
                )
                Select  [PD].[ItemKey]
                      , [PurchaseOrder] = [PD].[PURCHASEORDER]
                      , [Line] = ParseName(Replace([PD].[ItemKey] , '     ' ,
                                                   '.') , 1)
                      , [PM].[OrderEntryDate]
                      , [PM].[OrderDueDate]
                      , [PD].[DatabaseName]
                      , [PD].[TransactionDescription]
                      , [PD].[SignatureDateTime]
                      , [PD].[Operator]
                      , [PD].[PREVIOUSPRICE]
                      , [PD].[PRICE]
                      , [PD].[PREVIOUSFOREIGNPRICE]
                      , [PD].[FOREIGNPRICE]
                      , [PD].[PREVIOUSQUANTITY]
                      , [PD].[QUANTITY]
                      , [PD].[ORDERUNITOFMEASURE]
                      , [PriceDiff] = Abs([PD].[PREVIOUSPRICE] - [PD].[PRICE])
                      , [PriceDiffPercent] = Case When Coalesce([PD].[PREVIOUSPRICE]
,
                                                                0) = 0 Then 1
                                                 Else Abs([PD].[PREVIOUSPRICE]
                                                          - [PD].[PRICE])
                                                      / [PD].[PREVIOUSPRICE]
                                            End
                      , [ForeignPriceDiff] = Abs([PD].[PREVIOUSFOREIGNPRICE]
                                                 - [PD].[FOREIGNPRICE])
                      , [ForeignPriceDiffPercent] = Case When
Coalesce([PD].[PREVIOUSFOREIGNPRICE] ,
                                                                0) = 0 Then 1
                                                       Else
Abs([PD].[PREVIOUSFOREIGNPRICE]
                                                            - [PD].[FOREIGNPRICE])
                                                            /
[PD].[PREVIOUSFOREIGNPRICE]
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-PurchaseOrderChanges

```
                                                    End
                      , [QuantityDiff] = Abs([PD].[PREVIOUSQUANTITY]
                                             - [PD].[QUANTITY])
                      , [QuantityDiffPercent] = Case When
Coalesce([PD].[PREVIOUSQUANTITY] ,
                                                                0) = 0 Then 1
                                                    Else
Abs([PD].[PREVIOUSQUANTITY]
                                                                - [PD].[QUANTITY])
                                                           / [PD].[PREVIOUSQUANTITY]
                                                End
                      , [LineForeignValue] = ( [PD].[QUANTITY]
                                               * [PD].[FOREIGNPRICE] )
                      , [PrevLineForeignValue] = ( [PD].[PREVIOUSQUANTITY]
                                                   * [PD].[PREVIOUSFOREIGNPRICE] )
                      , [LineLocalValue] = ( [PD].[QUANTITY] * [PD].[PRICE] )
                      , [PrevLineLocalValue] = ( [PD].[PREVIOUSQUANTITY]
                                                 * [PD].[PREVIOUSPRICE] )
                      , [cn].[CompanyName]
                from    [History].[PorMasterDetail] [PD]
                        Inner Join [#PorMasterHdr] [PM] On [PD].[PURCHASEORDER] =
[PM].[PurchaseOrder] Collate Latin1_General_BIN
                                                           And [PM].[DatabaseName] =
[PD].[DatabaseName] Collate Latin1_General_BIN
                        Left Join [Lookups].[CompanyNames] As [cn] On 'Syspro-
Company'
                                                           + [cn].[Company] =
[PD].[DatabaseName] Collate Latin1_General_BIN
                Where   [PD].[TransactionDescription] In (
                        'PO Change purchase order merchandise line' )
                Order By [PM].[OrderEntryDate] Desc;



--return results

        Select  [ItemKey]
              , [PurchaseOrder]
              , [Line]
              , [OrderEntryDate] = Cast([OrderEntryDate] As Date)
              , [OrderDueDate] = Cast([OrderDueDate] As Date)
              , [Company] = Replace([DatabaseName] , 'SysproCompany' , '')
              , [CompanyName]
              , [SignatureDatetime]
              , [Operator]
              , [PreviousPrice]
              , [Price]
              , [PreviousForeignPrice]
              , [ForeignPrice]
              , [PreviousQuantity]
              , [Quantity]
              , [OrderUnitOfMeasure]
              , [PriceDiff]
              , [PriceDiffPercent]
              , [ForeignPriceDiff]
              , [ForeignPriceDiffPercent] = [ForeignPriceDiffPercent] * 100 --
crystal does not handle decimal percentages


              , [QuantityDiff]
```

Author: Johnson, Chris

```
              , [QuantityDiffPercent] = [QuantityDiffPercent] * 100
              , [LineForeignValue] = Coalesce([LineForeignValue] ,
                                              [ForeignPrice])
              , [PrevLineForeignValue] = Coalesce([PrevLineForeignValue] ,
                                                  [PreviousForeignPrice]) --take
into account POs without a quantity

              , [LineLocalValue] = Coalesce([LineLocalValue] , [Price])
              , [PrevLineLocalValue] = Coalesce([PrevLineLocalValue] ,
                                                [PreviousPrice])
        From    [#Results];

    End;


GO
EXEC sp_addextendedproperty N'MS_Description', N'list of changes made to purchase
orders', 'SCHEMA', N'Report', 'PROCEDURE', N'UspResults_PurchaseOrderChanges', NULL,
NULL
GO
```

Uses

[Lookups].[CompanyNames]
[Process].[ExecForEachDB]
[Process].[UspInsert_RedTagLogs]
[History].[PorMasterDetail]
[Report]

Author: Johnson, Chris

[Report].[UspResults_PurchaseOrderDetails]

MS_Description

purchase order details

Parameters

Name	Data Type	Max Length (Bytes)
@Company	varchar(max)	max
@RedTagType	char	1
@RedTagUse	varchar(500)	500

SQL Script

```
CREATE Proc [Report].[UspResults_PurchaseOrderDetails]
    (
      @Company Varchar(Max)
    , @RedTagType Char(1)
    , @RedTagUse Varchar(500)
    )
As
    Begin
/*
Template designed by Chris Johnson, Prometic Group September 2015
Stored procedure set out to query multiple databases with the same information and
return it in a collated format
--exec [Report].[UspResults_PurchaseOrderDetails] 43
*/
        If IsNumeric(@Company) = 0
            Begin
                Select  @Company = Upper(@Company);
            End;

--remove nocount on to speed up query

        Set NoCount On;
--Red tag

        Declare @RedTagDB Varchar(255)= Db_Name();
        Exec [Process].[UspInsert_RedTagLogs] @StoredProcDb = 'BlackBox' ,
            @StoredProcSchema = 'Report' ,
            @StoredProcName = 'UspResults_PurchaseOrderDetails' ,
            @UsedByType = @RedTagType , @UsedByName = @RedTagUse ,
            @UsedByDb = @RedTagDB;
--list the tables that are to be pulled back from each DB - if they are not found
the script will not be run against that db

        Declare @ListOfTables Varchar(Max) = 'PorMasterHdr,PorMasterDetail,Req-
Detail,ReqHeader,ApSupplier';

--create temporary tables to be pulled from different databases, including a column
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-PurchaseOrderDetails

```
to id

        Create Table [#PorMasterHdr]
            (
              [DatabaseName] Varchar(150)        Collate Latin1_General_BIN
            , [PurchaseOrder] Varchar(50)        Collate Latin1_General_BIN
            , [OrderEntryDate] DateTime2
            , [OrderDueDate] DateTime2
            , [Supplier] Varchar(50)            Collate Latin1_General_BIN
            , [DeliveryName] Varchar(100)        Collate Latin1_General_BIN
            , [DeliveryAddr1] Varchar(155)        Collate Latin1_General_BIN
            , [Currency] Varchar(10)            Collate Latin1_General_BIN
            , [OrderStatus] Varchar(10)            Collate Latin1_General_BIN
            );
        Create Table [#PorMasterDetail]
            (
              [DatabaseName] Varchar(150)        Collate Latin1_General_BIN
            , [PurchaseOrder] Varchar(50)        Collate Latin1_General_BIN
            , [Line] Int
            , [MStockCode] Varchar(50)            Collate Latin1_General_BIN
            , [MSupCatalogue] Varchar(150)        Collate Latin1_General_BIN
            , [MGlCode] Varchar(50)                Collate Latin1_General_BIN
            , [MStockDes] Varchar(150)            Collate Latin1_General_BIN
            , [MOrderUom] Varchar(10)            Collate Latin1_General_BIN
            , [MPrice] Numeric(20 , 3)
            , [MForeignPrice] Numeric(20 , 3)
            , [MOrderQty] Numeric(20 , 7)
            , [MRequisition] Varchar(50)        Collate Latin1_General_BIN
            , [MRequisitionLine] Int
            );
        Create Table [#ReqDetail]
            (
              [DatabaseName] Varchar(150)        Collate Latin1_General_BIN
            , [Buyer] Varchar(100)                Collate Latin1_General_BIN
            , [Originator] Varchar(100)            Collate Latin1_General_BIN
            , [ReasonForReqn] Varchar(500)        Collate Latin1_General_BIN
            , [Operator] Varchar(150)            Collate Latin1_General_BIN
            , [Requisition] Varchar(50)            Collate Latin1_General_BIN
            , [Line] Int
            );
        Create Table [#ReqHeader]
            (
              [DatabaseName] Varchar(150)        Collate Latin1_General_BIN
            , [DateReqnRaised] DateTime2
            , [Requisition] Varchar(50)            Collate Latin1_General_BIN
            );
        Create Table [#ApSupplier]
            (
              [DatabaseName] Varchar(150)        Collate Latin1_General_BIN
            , [Supplier] Varchar(50)            Collate Latin1_General_BIN
            , [SupplierName] Varchar(150)        Collate Latin1_General_BIN
            , [MerchGlCode] Varchar(50)            Collate Latin1_General_BIN
            );

--create script to pull data from each db into the tables
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-PurchaseOrderDetails

```
        Declare @SQL1 Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN'
            + --only companies selected in main run, or if companies selected then
all


            '
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT'
            + --count number of tables requested (number of commas plus one)

            '
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')'
            + --Count of the tables requested how many exist in the db

            '
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'','')) '
            + --only if the count matches (all the tables exist in the requested db)
then run the script

            '
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
                Insert [#PorMasterHdr]
                        ( [DatabaseName]
                        , [PurchaseOrder]
                        , [OrderEntryDate]
                        , [OrderDueDate]
                        , [Supplier]
                        , [DeliveryName]
                        , [DeliveryAddr1]
                        , [Currency]
                        ,[OrderStatus]
                        )
                SELECT [DatabaseName]=@DBCode
                     , [pmh].[PurchaseOrder]
                     , [pmh].[OrderEntryDate]
                     , [pmh].[OrderDueDate]
                     , [pmh].[Supplier]
                     , [pmh].[DeliveryName]
                     , [pmh].[DeliveryAddr1]
                     , [pmh].[Currency]
                     ,[OrderStatus]
                From [PorMasterHdr] As [pmh]
            End
    End';
        Declare @SQL2 Varchar(Max) = '
    USE [?];
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-PurchaseOrderDetails

```
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN'
            + --only companies selected in main run, or if companies selected then
all

            '
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT'
            + --count number of tables requested (number of commas plus one)

            '
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')'
            + --Count of the tables requested how many exist in the db

            '
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'','')) '
            + --only if the count matches (all the tables exist in the requested db)
then run the script

            '
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
                Insert [#PorMasterDetail]
                        ( [DatabaseName]
                        , [PurchaseOrder]
                        , [Line]
                        , [MStockCode]
                        , [MSupCatalogue]
                        , [MGlCode]
                        , [MStockDes]
                        , [MOrderUom]
                        , [MPrice]
                        , [MForeignPrice]
                        , [MOrderQty]
                        , [MRequisition]
                        , [MRequisitionLine]
                        )
                SELECT [DatabaseName]=@DBCode
                     , [pmd].[PurchaseOrder]
                     , [pmd].[Line]
                     , [pmd].[MStockCode]
                     , [pmd].[MSupCatalogue]
                     , [pmd].[MGlCode]
                     , [pmd].[MStockDes]
                     , [pmd].[MOrderUom]
                     , [pmd].[MPrice]
                     , [pmd].[MForeignPrice]
                     , [pmd].[MOrderQty]
                     , [pmd].[MRequisition]
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-PurchaseOrderDetails

```
                    , [pmd].[MRequisitionLine]
                FROM [PorMasterDetail] As [pmd]
            End
    End';
        Declare @SQL3 Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN'
            + --only companies selected in main run, or if companies selected then
all

            '
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT'
            + --count number of tables requested (number of commas plus one)

            '
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')'
            + --Count of the tables requested how many exist in the db

            '
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'',''))  '
            + --only if the count matches (all the tables exist in the requested db)
then run the script

            '
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
                Insert [#ReqDetail]
                        ( [DatabaseName]
                        , [Buyer]
                        , [Originator]
                        , [ReasonForReqn]
                        , [Operator]
                        , [Requisition]
                        , [Line]
                        )
                SELECT [DatabaseName]=@DBCode
                     , [rd].[Buyer]
                     , [rd].[Originator]
                     , [rd].[ReasonForReqn]
                     , [rd].[Operator]
                     , [rd].[Requisition]
                     , [rd].[Line]
                FROM [ReqDetail] As [rd]
            End
    End';
        Declare @SQL4 Varchar(Max) = '
    USE [?];
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-PurchaseOrderDetails

```
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN'
            + --only companies selected in main run, or if companies selected then
all

            '
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT'
            + --count number of tables requested (number of commas plus one)

            '
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')'
            + --Count of the tables requested how many exist in the db

            '
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'','')) '
            + --only if the count matches (all the tables exist in the requested db)
then run the script

            '
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
                Insert [#ReqHeader]
                        ( [DatabaseName]
                        , [DateReqnRaised]
                        , [Requisition]
                        )
                SELECT [DatabaseName]=@DBCode
                     , [rh].[DateReqnRaised]
                     , [rh].[Requisition]
                FROM [ReqHeader] As [rh]
            End
    End';
        Declare @SQL5 Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN'
            + --only companies selected in main run, or if companies selected then
all

            '
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT'
            + --count number of tables requested (number of commas plus one)
```

Author: Johnson, Chris

```
            '
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')'
            + --Count of the tables requested how many exist in the db

            '
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'','')) '
            + --only if the count matches (all the tables exist in the requested db)
then run the script

            '
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
                Insert [#ApSupplier]
                        ( [DatabaseName]
                        , [Supplier]
                        , [SupplierName]
                        , [MerchGlCode]
                        )
                SELECT [DatabaseName]=@DBCode
                     , [as].[Supplier]
                     , [as].[SupplierName]
                     , [as].[MerchGlCode]
                From [ApSupplier] As [as]
            End
    End';
--Enable this function to check script changes (try to run script directly against
db manually)

--Print @SQL


--execute script against each db, populating the base tables

        Exec [Process].[ExecForEachDB] @cmd = @SQL1;
        Exec [Process].[ExecForEachDB] @cmd = @SQL2;
        Exec [Process].[ExecForEachDB] @cmd = @SQL3;
        Exec [Process].[ExecForEachDB] @cmd = @SQL4;
        Exec [Process].[ExecForEachDB] @cmd = @SQL5;

--define the results you want to return

        Create Table [#PurchaseOrderDetailsResults]
            (
              [PurchaseOrder] Varchar(50)        collate Latin1_General_BIN
            , [OrderEntryDate] DateTime2
            , [OrderDueDate] DateTime2
            , [Supplier] Varchar(50)            collate Latin1_General_BIN
            , [DeliveryName] Varchar(50)        collate Latin1_General_BIN
            , [DeliveryAddr1] Varchar(150)        collate Latin1_General_BIN
            , [Currency] Varchar(5)               collate Latin1_General_BIN
            , [DateReqnRaised] DateTime
            , [Requisition] Varchar(50)           collate Latin1_General_BIN
            , [Line] Int
            , [StockCode] Varchar(50)           collate Latin1_General_BIN
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-PurchaseOrderDetails

```
            , [SupCatalogue] Varchar(150)        collate Latin1_General_BIN
            , [GlCode] Varchar(50)               collate Latin1_General_BIN
            , [StockDes] Varchar(150)            collate Latin1_General_BIN
            , [OrderUom] Varchar(10)             collate Latin1_General_BIN
            , [Price] Numeric(20 , 3)
            , [ForeignPrice] Numeric(20 , 3)
            , [OrderQty] Numeric(20 , 7)
            , [Buyer] Varchar(50)                collate Latin1_General_BIN
            , [Originator] Varchar(150)          collate Latin1_General_BIN
            , [ReasonForReqn] Varchar(255)       collate Latin1_General_BIN
            , [ReqOperator] Varchar(150)         collate Latin1_General_BIN
            , [SupplierName] Varchar(150)        collate Latin1_General_BIN
            , [MerchGlCode] Varchar(50)          collate Latin1_General_BIN
            , [CompanyName] Varchar(150)         collate Latin1_General_BIN
            , [OrderStatus] Varchar(250)         collate Latin1_General_BIN
            );

--Placeholder to create indexes as required


--script to combine base data and insert into results table

        Insert  [#PurchaseOrderDetailsResults]
                ( [PurchaseOrder]
                , [OrderEntryDate]
                , [OrderDueDate]
                , [Supplier]
                , [DeliveryName]
                , [DeliveryAddr1]
                , [Currency]
                , [DateReqnRaised]
                , [Requisition]
                , [Line]
                , [StockCode]
                , [SupCatalogue]
                , [GlCode]
                , [StockDes]
                , [OrderUom]
                , [Price]
                , [ForeignPrice]
                , [OrderQty]
                , [Buyer]
                , [Originator]
                , [ReasonForReqn]
                , [ReqOperator]
                , [SupplierName]
                , [MerchGlCode]
                , [CompanyName]
                , [OrderStatus]
                )
                Select  [pmh].[PurchaseOrder]
                      , [pmh].[OrderEntryDate]
                      , [pmh].[OrderDueDate]
                      , [pmh].[Supplier]
                      , [pmh].[DeliveryName]
                      , [pmh].[DeliveryAddr1]
                      , [pmh].[Currency]
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-PurchaseOrderDetails

```
                        , [rh].[DateReqnRaised]
                        , [rh].[Requisition]
                        , [pmd].[Line]
                        , [pmd].[MStockCode]
                        , [pmd].[MSupCatalogue]
                        , [pmd].[MGlCode]
                        , [pmd].[MStockDes]
                        , [pmd].[MOrderUom]
                        , [pmd].[MPrice]
                        , [pmd].[MForeignPrice]
                        , [pmd].[MOrderQty]
                        , [rd].[Buyer]
                        , [rd].[Originator]
                        , [rd].[ReasonForReqn]
                        , [ReqOperator] = Replace([rd].[Operator] , '.' , ' ')
                        , [as].[SupplierName]
                        , [as].[MerchGlCode]
                        , [cn].[CompanyName]
                        , [pos].[OrderStatusDescription]
                From    [#PorMasterHdr] As [pmh]
                        Left Join [#PorMasterDetail] As [pmd] On [pmd].[Purchase-
Order] = [pmh].[PurchaseOrder]
                        Left Join [#ReqDetail] As [rd] On [pmd].[MRequisition] =
[rd].[Requisition]
                                                           And [pmd].[MRequisition-
Line] = [rd].[Line]
                        Left Join [#ReqHeader] As [rh] On [rh].[Requisition] =
[rd].[Requisition]
                        Left Join [#ApSupplier] As [as] On [as].[Supplier] =
[pmh].[Supplier]
                        Left Join [BlackBox].[Lookups].[CompanyNames] As [cn] On
[cn].[Company] = [pmh].[DatabaseName]
                        Left Join [BlackBox].[Lookups].[PurchaseOrderStatus]
                        As [pos] On [pos].[Company] = [pmh].[DatabaseName]
                                    And [pos].[OrderStatusCode] = [pmh].[Order-
Status];



--return results

        Select  [PurchaseOrder]
              , [OrderEntryDate] = Cast([OrderEntryDate] As Date)
              , [OrderDueDate] = Cast([OrderDueDate] As Date)
              , [Supplier]
              , [DeliveryName]
              , [DeliveryAddr1]
              , [Currency]
              , [DateReqnRaised] = Cast([DateReqnRaised] As Date)
              , [Requisition]
              , [Line]
              , [StockCode]
              , [SupCatalogue]
              , [GlCode]
              , [StockDes]
              , [OrderUom]
              , [Price]
              , [ForeignPrice]
              , [OrderQty]
```

Author: Johnson, Chris

```
                , [Buyer]
                , [Originator]
                , [ReasonForReqn]
                , [ReqOperator]
                , [SupplierName]
                , [MerchGlCode]
                , [CompanyName]
                , [OrderStatus]
        From    [#PurchaseOrderDetailsResults];

--Tidy up

        Drop Table [#ApSupplier];
        Drop Table [#PorMasterDetail];
        Drop Table [#PorMasterHdr];
        Drop Table [#ReqDetail];
        Drop Table [#ReqHeader];
        Drop Table [#PurchaseOrderDetailsResults];

    End;

GO
EXEC sp_addextendedproperty N'MS_Description', N'purchase order details', 'SCHEMA',
N'Report', 'PROCEDURE', N'UspResults_PurchaseOrderDetails', NULL, NULL
GO
```

Uses

[Lookups].[CompanyNames]
[Lookups].[PurchaseOrderStatus]
[Process].[ExecForEachDB]
[Process].[UspInsert_RedTagLogs]
[Report]

Author: Johnson, Chris

[Report].[UspResults_PurchaseOrders]

MS_Description

purchase order details

Parameters

Name	Data Type	Max Length (Bytes)
@Company	varchar(max)	max
@RedTagType	char	1
@RedTagUse	varchar(500)	500

SQL Script

```
CREATE Proc [Report].[UspResults_PurchaseOrders]
    (
      @Company Varchar(Max)
    , @RedTagType Char(1)
    , @RedTagUse Varchar(500)
    )
As
    Begin
/*
Template designed by Chris Johnson, Prometic Group September 2015
Stored procedure set out to query multiple databases with the same information and
return it in a collated format
Procedure to return all Purchase Order Details
*/
        If IsNumeric(@Company) = 0
            Begin
                Select  @Company = Upper(@Company);
            End;

--remove nocount on to speed up query

        Set NoCount On;

--Red tag

        Declare @RedTagDB Varchar(255)= Db_Name();
        Exec [Process].[UspInsert_RedTagLogs] @StoredProcDb = 'BlackBox' ,
            @StoredProcSchema = 'Report' ,
            @StoredProcName = 'UspResults_PurchaseOrders' ,
            @UsedByType = @RedTagType , @UsedByName = @RedTagUse ,
            @UsedByDb = @RedTagDB;

--list the tables that are to be pulled back from each DB - if they are not found
the script will not be run against that db

        Declare @ListOfTables Varchar(Max) = 'InvBuyer,PorMasterHdr,PorMaster-
Detail';
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-PurchaseOrders

```
--create temporary tables to be pulled from different databases, including a column
to id

        Create Table [#Buyers]
            (
              [DatabaseName] Varchar(150) Collate Latin1_General_BIN
            , [BuyerCode] Varchar(20) Collate Latin1_General_BIN
            , [BuyerName] Varchar(50) Collate Latin1_General_BIN
            );

        Create Table [#PurchaseHeader]
            (
              [DatabaseName] Varchar(150) Collate Latin1_General_BIN
            , [OrderStatus] Char(1) Collate Latin1_General_BIN
            , [PurchaseOrder] Varchar(20) Collate Latin1_General_BIN
            , [ExchRateFixed] Char(1) Collate Latin1_General_BIN
            , [Currency] Char(3) Collate Latin1_General_BIN
            , [ExchangeRate] Float
            , [OrderType] Char(1) Collate Latin1_General_BIN
            , [TaxStatus] Char(1) Collate Latin1_General_BIN
            , [Customer] Varchar(15) Collate Latin1_General_BIN
            , [PaymentTerms] Char(3) Collate Latin1_General_BIN
            , [Buyer] Varchar(20) Collate Latin1_General_BIN
            , [ShippingInstrs] Varchar(50) Collate Latin1_General_BIN
            , [ShippingLocation] Varchar(10) Collate Latin1_General_BIN
            , [SupplierClass] Varchar(10) Collate Latin1_General_BIN
            , [Supplier] Varchar(15) Collate Latin1_General_BIN
            , [OrderEntryDate] DateTime2
            , [OrderDueDate] DateTime2
            , [DateLastDocPrt] DateTime2
            , [DatePoCompleted] DateTime2
            , [EdiPoFlag] Char(1) Collate Latin1_General_BIN
            , [EdiExtractFlag] Char(1) Collate Latin1_General_BIN
            , [EdiActionFlag] Char(1) Collate Latin1_General_BIN
            , [EdiConfirmation] Char(1) Collate Latin1_General_BIN
            );

        Create Table [#PurchaseDetails]
            (
              [DatabaseName] Varchar(150) Collate Latin1_General_BIN
            , [PurchaseOrder] Varchar(20) Collate Latin1_General_BIN
            , [Line] Decimal
            , [LineType] Char(1) Collate Latin1_General_BIN
            , [MStockCode] Varchar(30) Collate Latin1_General_BIN
            , [MStockDes] Varchar(50) Collate Latin1_General_BIN
            , [MWarehouse] Varchar(10) Collate Latin1_General_BIN
            , [MOrderUom] Varchar(10) Collate Latin1_General_BIN
            , [MStockingUom] Varchar(10) Collate Latin1_General_BIN
            , [MOrderQty] Float
            , [MReceivedQty] Float
            , [MLatestDueDate] DateTime2
            , [MLastReceiptDate] Date
            , [MPrice] Float--DECIMAL(15,12)

            , [MForeignPrice] Float--DECIMAL(15,12)
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-PurchaseOrders

```
            , [MDecimalsToPrt] Int
            , [MPriceUom] Varchar(10) Collate Latin1_General_BIN
            , [MTaxCode] Char(3) Collate Latin1_General_BIN
            , [MProductClass] Varchar(20) Collate Latin1_General_BIN
            , [MCompleteFlag] Char(1) Collate Latin1_General_BIN
            , [MJob] Varchar(20) Collate Latin1_General_BIN
            , [MJobLine] Char(2) Collate Latin1_General_BIN
            , [MGlCode] Varchar(35) Collate Latin1_General_BIN
            , [MUserAuthReqn] Varchar(20) Collate Latin1_General_BIN
            , [MRequisition] Varchar(10) Collate Latin1_General_BIN
            , [MRequisitionLine] Decimal
            , [MSalesOrder] Varchar(20) Collate Latin1_General_BIN
            , [MSalesOrderLine] Decimal
            , [MOrigDueDate] DateTime2
            , [MReschedDueDate] Char(1) Collate Latin1_General_BIN
            , [MSubcontractOp] Decimal
            , [MInspectionReqd] Char(1) Collate Latin1_General_BIN
            , [NMscChargeValue] Float
            , [CapexCode] Varchar(15) Collate Latin1_General_BIN
            , [CapexLine] Decimal
            , [NComment] Varchar(100) Collate Latin1_General_BIN
            );




--create script to pull data from each db into the tables

        Declare @SQL1 Varchar(Max) = '
USE [?];
Declare @DB varchar(150),@DBCode varchar(150)
Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end
IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
BEGIN'
            + --only companies selected in main run, or if companies selected then
all

            '
    IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
        Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables + '''
                , @RequiredCountOfTables INT
                , @ActualCountOfTables INT'
            + --count number of tables requested (number of commas plus one)

            '
        Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')'
            + --Count of the tables requested how many exist in the db

            '
        Select @ActualCountOfTables = COUNT(1) FROM sys.tables
        Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'',''))  '
            + --only if the count matches (all the tables exist in the requested db)
then run the script
```

Author: Johnson, Chris

```
            '
        If @ActualCountOfTables=@RequiredCountOfTables
        BEGIN
        Insert #Buyers
                (BuyerCode, BuyerName, DatabaseName
                )
                Select [Buyer], [Name], DatabaseName = @DBCode
                From dbo.InvBuyer
        End
End';
        Declare @SQL2 Varchar(Max) = '
USE [?];
Declare @DB varchar(150),@DBCode varchar(150)
Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end'
            + --Only query DBs beginning SysProCompany

            '
IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
BEGIN'
            + --only companies selected in main run, or if companies selected then
all

            '
    IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
        Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables + '''
                , @RequiredCountOfTables INT
                , @ActualCountOfTables INT'
            + --count number of tables requested (number of commas plus one)

            '
        Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')'
            + --Count of the tables requested how many exist in the db

            '
        Select @ActualCountOfTables = COUNT(1) FROM sys.tables
        Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'',''))  '
            + --only if the count matches (all the tables exist in the requested db)
then run the script

            '
        If @ActualCountOfTables=@RequiredCountOfTables
        BEGIN
        Insert #PurchaseHeader
                    ( DatabaseName, OrderStatus, PurchaseOrder
                    , ExchRateFixed, Currency, ExchangeRate
                    , OrderType, TaxStatus, Customer
                    , PaymentTerms, Buyer, ShippingInstrs
                    , ShippingLocation, SupplierClass, Supplier
                    , OrderEntryDate, OrderDueDate, DateLastDocPrt
                    , DatePoCompleted, EdiPoFlag, EdiExtractFlag
                    , EdiActionFlag, EdiConfirmation
                    )
                    SELECT
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-PurchaseOrders

```
                        DatabaseName = @DBCode, OrderStatus, PurchaseOrder
                        ,ExchRateFixed, Currency, ExchangeRate
                        ,OrderType, TaxStatus, Customer
                        ,PaymentTerms, Buyer, ShippingInstrs
                        ,ShippingLocation, SupplierClass, Supplier
                        ,OrderEntryDate, OrderDueDate, DateLastDocPrt
                        ,DatePoCompleted, EdiPoFlag, EdiExtractFlag
                        ,EdiActionFlag, EdiConfirmation
                FROM PorMasterHdr
        End
End';
        Declare @SQL3 Varchar(Max) = '
USE [?];
Declare @DB varchar(150),@DBCode varchar(150)
Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end'
            + --Only query DBs beginning SysProCompany

            '
IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
BEGIN'
            + --only companies selected in main run, or if companies selected then
all

            '
    IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
        Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables + '''
                , @RequiredCountOfTables INT
                , @ActualCountOfTables INT'
            + --count number of tables requested (number of commas plus one)

            '
        Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')'
            + --Count of the tables requested how many exist in the db

            '
        Select @ActualCountOfTables = COUNT(1) FROM sys.tables
        Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'',''))  '
            + --only if the count matches (all the tables exist in the requested db)
then run the script

            '
        If @ActualCountOfTables=@RequiredCountOfTables
        BEGIN
        Insert #PurchaseDetails
                ( DatabaseName, PurchaseOrder, Line
                , LineType, MStockCode, MStockDes
                , MWarehouse, MOrderUom, MStockingUom
                , MOrderQty, MReceivedQty, MLatestDueDate
                , MLastReceiptDate, MPrice, MForeignPrice
                , MDecimalsToPrt, MPriceUom, MTaxCode
                , MProductClass, MCompleteFlag, MJob
                , MJobLine, MGlCode, MUserAuthReqn
                , MRequisition, MRequisitionLine, MSalesOrder
                , MSalesOrderLine, MOrigDueDate, MReschedDueDate
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-PurchaseOrders

```
                , MSubcontractOp, MInspectionReqd, NMscChargeValue
                , CapexCode, CapexLine, NComment
                )
            SELECT
            DatabaseName = @DBCode, PurchaseOrder, Line
                , LineType, MStockCode, MStockDes
                , MWarehouse, MOrderUom, MStockingUom
                , MOrderQty, MReceivedQty, MLatestDueDate
                , MLastReceiptDat, MPrice, MForeignPrice
                , MDecimalsToPrt, MPriceUom, MTaxCode
                , MProductClass, MCompleteFlag, MJob
                , MJobLine, MGlCode, MUserAuthReqn
                , MRequisition, MRequisitionLine, MSalesOrder
                , MSalesOrderLine, MOrigDueDate
                , MReschedDueDate, MSubcontractOp
                , MInspectionReqd, NMscChargeValue
                , CapexCode, CapexLine, NComment
        From PorMasterDetail

        End
End';


--Enable this function to check script changes (try to run script directly against
db manually)

--Print @SQL1

--Print @SQL2

--Print @SQL3



--execute script against each db, populating the base tables

        Exec [Process].[ExecForEachDB] @cmd = @SQL1;
        Exec [Process].[ExecForEachDB] @cmd = @SQL2;
        Exec [Process].[ExecForEachDB] @cmd = @SQL3;

--define the results you want to return


--Placeholder to create indexes as required

        Create NonClustered Index [DiX_Buyers_DBName] On [#Buyers] ([Database-
Name],[BuyerCode]);
        Create NonClustered Index [DiX_PH_DBName] On [#PurchaseHeader] ([Database-
Name],[PurchaseOrder],[Buyer]);
        Create NonClustered Index [DiX_PD_DBName] On [#PurchaseDetails] ([Database-
Name],[PurchaseOrder]);

--script to combine base data and insert into results table

        Select  [Company] = [PH].[DatabaseName]
              , [BuyerName] = Coalesce([B].[BuyerName] , [PH].[Buyer])
              , [OrderStatus] = Coalesce([PS].[OrderStatusDescription]  Collate
Latin1_General_BIN ,
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-PurchaseOrders

```
                                        [PH].[OrderStatus])
              , [PH].[PurchaseOrder]
              , [PH].[ExchRateFixed]
              , [PH].[Currency]
              , [PH].[ExchangeRate]
              , [OrderType] = Coalesce([PT].[OrderTypeDescription] Collate Latin1_-
General_BIN ,
                                       [PH].[OrderType])
              , [TaxStatus] = Coalesce([POTS].[TaxStatusDescription] Collate Latin1_-
General_BIN ,
                                       [PH].[TaxStatus])
              , [PH].[Customer]
              , [PH].[PaymentTerms]
              , [PH].[ShippingInstrs]
              , [PH].[ShippingLocation]
              , [PH].[SupplierClass]
              , [PH].[Supplier]
              , [OrderEntryDate] = Cast([PH].[OrderEntryDate] As Date)
              , [OrderDueDate] = Cast([PH].[OrderDueDate] As Date)
              , [PH].[DateLastDocPrt]
              , [PH].[DatePoCompleted]
              , [PH].[EdiPoFlag]
              , [PH].[EdiExtractFlag]
              , [PH].[EdiActionFlag]
              , [PH].[EdiConfirmation]
              , [PD].[Line]
              , [PD].[LineType]
              , [PD].[MStockCode]
              , [PD].[MStockDes]
              , [PD].[MWarehouse]
              , [PD].[MOrderUom]
              , [PD].[MStockingUom]
              , [MOrderQty] = Coalesce([PD].[MOrderQty] , 0)
              , [PD].[MReceivedQty]
              , [PD].[MLatestDueDate]
              , [PD].[MLastReceiptDate]
              , [MPrice] = Coalesce([PD].[MPrice] , 0)
              , [MForeignPrice] = Coalesce([PD].[MForeignPrice] , 0)
              , [PD].[MDecimalsToPrt]
              , [PD].[MPriceUom]
              , [PD].[MTaxCode]
              , [ProductClass] = Coalesce([PC].[ProductClassDescription]  Collate
Latin1_General_BIN ,
                                      Case When [PD].[MProductClass] = ''
                                           Then 'No Class'
                                           Else [PD].[MProductClass]
                                      End)
              , [MCompleteFlag] = Coalesce([MC].[MCompleteFlagDescription] Collate
Latin1_General_BIN ,
                                       [PD].[MCompleteFlag])
              , [PD].[MJob]
              , [PD].[MJobLine]
              , [PD].[MGlCode]
              , [PD].[MUserAuthReqn]
              , [PD].[MRequisition]
              , [PD].[MRequisitionLine]
              , [PD].[MSalesOrder]
              , [PD].[MSalesOrderLine]
```

Author: Johnson, Chris

```
                , [PD].[MOrigDueDate]
                , [PD].[MReschedDueDate]
                , [PD].[MSubcontractOp]
                , [PD].[MInspectionReqd]
                , [NMscChargeValue] = Coalesce([PD].[NMscChargeValue] , 0)
                , [PD].[CapexCode]
                , [PD].[CapexLine]
                , [PD].[NComment]
        --Into    [#Results]

        From    [#PurchaseHeader] [PH]
                Left Join [#Buyers] [B] On [B].[DatabaseName] = [PH].[DatabaseName]
                                        And [B].[BuyerCode] = [PH].[Buyer]
                Left Join [#PurchaseDetails] [PD] On [PD].[DatabaseName] =
[PH].[DatabaseName]
                                                And [PD].[PurchaseOrder] =
[PH].[PurchaseOrder]
                Left Join [Lookups].[PurchaseOrderStatus] [PS] On [PS].[Company] =
[PH].[DatabaseName] Collate Latin1_General_BIN
                                                        And [PH].[OrderStatus]
= [PS].[OrderStatusCode] Collate Latin1_General_BIN
                Left Join [Lookups].[PurchaseOrderType] [PT] On [PT].[Company] =
[PH].[DatabaseName] Collate Latin1_General_BIN
                                                        And [PT].[OrderType-
Code] = [PH].[OrderType] Collate Latin1_General_BIN
                Left Join [Lookups].[PurchaseOrderTaxStatus] [POTS] On
[POTS].[Company] = [PH].[DatabaseName] Collate Latin1_General_BIN
                                                        And [POTS].[TaxStatus-
Code] = [PH].[TaxStatus] Collate Latin1_General_BIN
                Left Join [Lookups].[MCompleteFlag] [MC] On [MC].[Company] =
[PD].[DatabaseName] Collate Latin1_General_BIN
                                                      And [MC].[MCompleteFlag-
Code] = [PD].[MCompleteFlag] Collate Latin1_General_BIN
                Left Join [Lookups].[ProductClass] [PC] On [PC].[Company] =
[PH].[DatabaseName] Collate Latin1_General_BIN
                                                    And [PC].[ProductClass] =
[PD].[MProductClass] Collate Latin1_General_BIN;

--return results


    End;


GO
EXEC sp_addextendedproperty N'MS_Description', N'purchase order details', 'SCHEMA',
N'Report', 'PROCEDURE', N'UspResults_PurchaseOrders', NULL, NULL
GO
```

Uses

[Lookups].[MCompleteFlag]
[Lookups].[ProductClass]
[Lookups].[PurchaseOrderStatus]
[Lookups].[PurchaseOrderTaxStatus]
[Lookups].[PurchaseOrderType]
[Process].[ExecForEachDB]
[Process].[UspInsert_RedTagLogs]

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-PurchaseOrders

[Report]

Author: Johnson, Chris

[Report].[UspResults_PurchaseOrdersHistory]

MS_Description

list of changes made to purchase orders

Parameters

Name	Data Type	Max Length (Bytes)
@Company	varchar(10)	10
@RedTagType	char	1
@RedTagUse	varchar(500)	500

SQL Script

```
CREATE Proc [Report].[UspResults_PurchaseOrdersHistory] ( @Company Varchar(10)      ,
@RedTagType Char(1)
    , @RedTagUse Varchar(500)
    )
As
    Begin
/*
Template designed by Chris Johnson, Prometic Group September 2015
Stored procedure set out to query multiple databases with the same information and
return it in a collated format
Procedure to return all Purchase Order Details and changes
--Exec [Report].[UspResults_PurchaseOrdersHistory] 10
*/
        If IsNumeric(@Company) = 0
            Begin
                Select  @Company = Upper(@Company);
            End;

--remove nocount on to speed up query

        Set NoCount On;

--Red tag

        Declare @RedTagDB Varchar(255)= Db_Name();
        Exec [Process].[UspInsert_RedTagLogs] @StoredProcDb = 'BlackBox' ,
            @StoredProcSchema = 'Report' ,
            @StoredProcName = 'UspResults_PurchaseOrdersHistory' ,
            @UsedByType = @RedTagType , @UsedByName = @RedTagUse ,
            @UsedByDb = @RedTagDB;

--list the tables that are to be pulled back from each DB - if they are not found
the script will not be run against that db

        Declare @ListOfTables Varchar(Max) = 'PorMasterDetail';

--create temporary tables to be pulled from different databases, including a column
```

Author: Johnson, Chris

```
to id

        Create Table [#PorMasterDetail]
            (
              [DatabaseName] Varchar(150)        Collate Latin1_General_BIN
            , [DatabaseCode] Varchar(150)        Collate Latin1_General_BIN
            , [PurchaseOrder] Varchar(35)        Collate Latin1_General_BIN
            , [Line] Int
            , [StockCode] Varchar(35)            Collate Latin1_General_BIN
            , [StockDescription] Varchar(255)    Collate Latin1_General_BIN
            , [MStockingUom] Varchar(10)        Collate Latin1_General_BIN
            , [MOrderQty] Numeric(20 , 7)
            , [MReceivedQty] Numeric(20 , 7)
            , [MLatestDueDate] DateTime2
            , [MOrigDueDate] DateTime2
            , [MPrice] Numeric(20 , 3)
            , [MForeignPrice] Numeric(20 , 3)
            );

--create script to pull data from each db into the tables

        Declare @SQL1 Varchar(Max) = '
USE [?];
Declare @DB varchar(150),@DBCode varchar(150)
Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end
IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
BEGIN'
            + --only companies selected in main run, or if companies selected then
all

            '
    IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
        Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables + '''
                , @RequiredCountOfTables INT
                , @ActualCountOfTables INT'
            + --count number of tables requested (number of commas plus one)

            '
        Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')'
            + --Count of the tables requested how many exist in the db

            '
        Select @ActualCountOfTables = COUNT(1) FROM sys.tables
        Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'',''))  '
            + --only if the count matches (all the tables exist in the requested db)
then run the script

            '
        If @ActualCountOfTables=@RequiredCountOfTables
        BEGIN
        Insert [#PorMasterDetail]
                ( [DatabaseName]
                , [DatabaseCode]
                , [PurchaseOrder]
```

Author: Johnson, Chris

```
                , [Line]
                , [StockCode]
                , [StockDescription]
                , [MStockingUom]
                , [MOrderQty]
                , [MReceivedQty]
                , [MLatestDueDate]
                , [MOrigDueDate]
                , [MPrice]
                , [MForeignPrice]
                )
        SELECT [DatabaseName]=@DB
             , [DatabaseCode]=@DBCode
             , [pmd].[PurchaseOrder]
             , [pmd].[Line]
             , [pmd].[MStockCode]
             , [pmd].[MStockDes]
             , [pmd].[MStockingUom]
             , [pmd].[MOrderQty]
             , [pmd].[MReceivedQty]
             , [pmd].[MLatestDueDate]
             , [pmd].[MOrigDueDate]
             , [pmd].[MPrice]
             , [pmd].[MForeignPrice]
        From [PorMasterDetail] As [pmd]
        End
End';

--Enable this function to check script changes (try to run script directly against
db manually)

--Print @SQL1

--Print @SQL2

--Print @SQL3


--execute script against each db, populating the base tables

        Exec [Process].[ExecForEachDB] @cmd = @SQL1;

--define the results you want to return

        Create Table [#Results]
            (
              [CompanyName] Varchar(150)                    Collate Latin1_General_-
BIN
            , [PurchaseOrder] Varchar(35)                    collate Latin1_General_-
BIN
            , [Line] Int
            , [StockCode] Varchar(35)                       collate Latin1_General_-
BIN
            , [StockDescription] Varchar(255)               collate Latin1_General_-
BIN
            , [StockingUom] Varchar(10)                       collate Latin1_-
General_BIN
            , [OrderQty] Numeric(20 , 7)
```

Author: Johnson, Chris

```
            , [ReceivedQty] Numeric(20 , 7)
            , [LatestDueDate] DateTime2
            , [OrigDueDate] DateTime2
            , [LocalPrice] Numeric(20 , 3)
            , [ForeignPrice] Numeric(20 , 3)
            , [TransactionDescription] Varchar(255)           collate Latin1_-
General_BIN
            , [SignatureDatetime] DateTime2
            , [Operator] Varchar(255)                       collate Latin1_General_-
BIN
            , [Price] Numeric(20 , 3)
            , [PreviousPrice] Numeric(20 , 3)
            , [Quantity] Numeric(20 , 7)
            , [PreviousQuantity] Numeric(20 , 7)
            , [QuantityBeingReceieved] Numeric(20 , 7)
            , [Grn] Varchar(50)                               collate Latin1_-
General_BIN
            );

--Placeholder to create indexes as required



--script to combine base data and insert into results table

        Insert  [#Results]
                ( [PurchaseOrder]
                , [Line]
                , [StockCode]
                , [StockDescription]
                , [StockingUom]
                , [OrderQty]
                , [ReceivedQty]
                , [LatestDueDate]
                , [OrigDueDate]
                , [LocalPrice]
                , [ForeignPrice]
                , [TransactionDescription]
                , [SignatureDatetime]
                , [Operator]
                , [Price]
                , [PreviousPrice]
                , [Quantity]
                , [PreviousQuantity]
                , [QuantityBeingReceieved]
                , [Grn]
                , [CompanyName]
                )
                Select  [PurchaseOrder] = Coalesce([md].[PurchaseOrder] ,
                                                   [pmd].[PURCHASEORDER])
                      , [Line] = Coalesce([md].[Line] ,
                                          [pmd].[PURCHASEORDERLINE])
                      , [StockCode] = Coalesce([md].[StockCode] ,
                                               [pmd].[STOCKCODE])
                      , [StockDescription] = Coalesce([md].[StockDescription] ,
                                                      [pmd].[STOCKDESCRIPTION])
                      , [md].[MStockingUom]
                      , [md].[MOrderQty]
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-PurchaseOrdersHistory

```
                      , [md].[MReceivedQty]
                      , [md].[MLatestDueDate]
                      , [md].[MOrigDueDate]
                      , [md].[MPrice]
                      , [md].[MForeignPrice]
                      , [pmd].[TransactionDescription]
                      , [pmd].[SignatureDateTime]
                      , [pmd].[Operator]
                      , [pmd].[PRICE]
                      , [pmd].[PREVIOUSPRICE]
                      , [pmd].[QUANTITY]
                      , [pmd].[PREVIOUSQUANTITY]
                      , [pmd].[QUANTITYBEINGRECEIVED]
                      , [Grn] = [pmd].[GOODSRECEIVEDNUMBER]
                      , [cn].[CompanyName]
                From    [#PorMasterDetail] As [md]
                        Inner Join [BlackBox].[History].[PorMasterDetail] As [pmd]
On [pmd].[PURCHASEORDER] = [md].[PurchaseOrder]
                                                                And [md].[Line] =
[pmd].[PURCHASEORDERLINE]
                                                                And [pmd].[Database-
Name] = [md].[DatabaseName]
                        Left Join [Lookups].[CompanyNames] As [cn] On [cn].[Company]
= [md].[DatabaseCode]
                Order By [md].[PurchaseOrder] Asc
                      , [md].[Line] Asc
                      , [pmd].[SignatureDateTime] Desc;

--return results

        Select  [CompanyName]
              , [PurchaseOrder]
              , [Line]
              , [StockCode]
              , [StockDescription]
              , [StockingUom]
              , [OrderQty]
              , [ReceivedQty]
              , [LatestDueDate] = Cast([LatestDueDate] As Date)
              , [OrigDueDate] = Cast([OrigDueDate] As Date)
              , [LocalPrice]
              , [ForeignPrice]
              , [TransactionDescription]
              , [SignatureDate] = Cast([SignatureDatetime] As Date)
              , [SignatureTime] = Cast([SignatureDatetime] As Time)
              , [Operator]
              , [Price]
              , [PreviousPrice]
              , [Quantity]
              , [PreviousQuantity]
              , [QuantityBeingReceieved]
              , [Grn]
        From    [#Results];

    End;
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-PurchaseOrdersHistory

```
GO
EXEC sp_addextendedproperty N'MS_Description', N'list of changes made to purchase orders', 'SCHEMA', N'Report', 'PROCEDURE', N'UspResults_PurchaseOrdersHistory', NULL, NULL
GO
```

Uses

[Lookups].[CompanyNames]
[Process].[ExecForEachDB]
[Process].[UspInsert_RedTagLogs]
[History].[PorMasterDetail]
[Report]

Author: Johnson, Chris

[Report].[UspResults_PurchaseOrdersInvoices]

MS_Description

purchase order details with invoices

Parameters

Name	Data Type	Max Length (Bytes)
@Company	varchar(max)	max
@RedTagType	char	1
@RedTagUse	varchar(500)	500

SQL Script

```
CREATE Proc [Report].[UspResults_PurchaseOrdersInvoices]
    (
      @Company Varchar(Max)
    , @RedTagType Char(1)
    , @RedTagUse Varchar(500)
    )
As
    Begin
/*
Template designed by Chris Johnson, Prometic Group September 2015
Stored procedure set out to query multiple databases with the same information and
return it in a collated format
Return details of all purchase orders and invoices, highlighting where a PO is not
available or was entered late
*/
        If IsNumeric(@Company) = 0
            Begin
                Select  @Company = Upper(@Company);
            End;

--remove nocount on to speed up query

        Set NoCount On;

--Red tag

        Declare @RedTagDB Varchar(255)= Db_Name();
        Exec [Process].[UspInsert_RedTagLogs] @StoredProcDb = 'BlackBox' ,
            @StoredProcSchema = 'Report' ,
            @StoredProcName = 'UspResults_PurchaseOrdersInvoices' ,
            @UsedByType = @RedTagType , @UsedByName = @RedTagUse ,
            @UsedByDb = @RedTagDB;

--list the tables that are to be pulled back from each DB - if they are not found
the script will not be run against that db

        Declare @ListOfTables Varchar(Max) = 'CshApPayments,ApInvoice,ApInvoice-
```

Author: Johnson, Chris

```
Pay,PorMasterHdr,ApSupplier';

--create temporary tables to be pulled from different databases, including a column
to id

        Create Table [#CshApPayments]
            (
              [DatabaseName] Varchar(150) Collate Latin1_General_BIN
            , [CbTrnYear] Int
            , [TrnMonth] Int
            , [Supplier] Varchar(15) Collate Latin1_General_BIN
            , [Invoice] Varchar(35) Collate Latin1_General_BIN
            , [InvoiceDate] DateTime2
            , [Reference] Varchar(60) Collate Latin1_General_BIN
            , [PaymentNumber] Varchar(35) Collate Latin1_General_BIN
            );
        Create Table [#ApInvoice]
            (
              [DatabaseName] Varchar(150) Collate Latin1_General_BIN
            , [Supplier] Varchar(15) Collate Latin1_General_BIN
            , [Invoice] Varchar(35) Collate Latin1_General_BIN
            , [PaymentNumber] Varchar(35) Collate Latin1_General_BIN
            );
        Create Table [#ApInvoicePay]
            (
              [DatabaseName] Varchar(150) Collate Latin1_General_BIN
            , [TrnValue] Float
            , [PaymentReference] Varchar(35) Collate Latin1_General_BIN
            , [Supplier] Varchar(15) Collate Latin1_General_BIN
            , [Invoice] Varchar(35) Collate Latin1_General_BIN
            );
        Create Table [#PorMasterHdr]
            (
              [DatabaseName] Varchar(150) Collate Latin1_General_BIN
            , [OrderEntryDate] DateTime2
            , [PurchaseOrder] Varchar(35) Collate Latin1_General_BIN
            );
        Create Table [#ApSupplier]
            (
              [DatabaseName] Varchar(150) Collate Latin1_General_BIN
            , [Supplier] Varchar(35) Collate Latin1_General_BIN
            , [SupplierName] Varchar(150) Collate Latin1_General_BIN
            );


--create script to pull data from each db into the tables

        Declare @SQL1 Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end'
            + --Only query DBs beginning SysProCompany

            '
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN'
            + --only companies selected in main run, or if companies selected then
```

```
all

            '
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT'
            + --count number of tables requested (number of commas plus one)

            '
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')'
            + --Count of the tables requested how many exist in the db

            '
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'','')) '
            + --only if the count matches (all the tables exist in the requested db)
then run the script

            '
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
                        Insert  #CshApPayments
                                    ( DatabaseName
                                    , CbTrnYear
                                    , TrnMonth
                                    , Supplier
                                    , Invoice
                                    , InvoiceDate
                                    , Reference
                                    , PaymentNumber
                                    )
                                    Select
                                        [DatabaseName] = @DBCode
                                      , CbTrnYear
                                      , TrnMonth
                                      , Supplier
                                      , Invoice
                                      , InvoiceDate
                                      , Reference
                                      , PaymentNumber
                                    From
                                        CshApPayments;
            End
    End';
        Declare @SQL2 Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end'
            + --Only query DBs beginning SysProCompany

            '
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN'
```

Author: Johnson, Chris

```
            + --only companies selected in main run, or if companies selected then
all

            '
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT'
            + --count number of tables requested (number of commas plus one)

            '
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')'
            + --Count of the tables requested how many exist in the db

            '
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'','')) '
            + --only if the count matches (all the tables exist in the requested db)
then run the script

            '
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
                 Insert  #ApInvoice
                            ( DatabaseName
                            , Supplier
                            , Invoice
                            , PaymentNumber
                            )
                            Select
                                [DatabaseName] = @DBCode
                              , Supplier
                              , Invoice
                              , PaymentNumber
                            From
                                ApInvoice;
            End
    End';
        Declare @SQL3 Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end'
            + --Only query DBs beginning SysProCompany

            '
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN'
            + --only companies selected in main run, or if companies selected then
all

            '
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables + '''
```

Author: Johnson, Chris

```
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT'
            + --count number of tables requested (number of commas plus one)

            '
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')'
            + --Count of the tables requested how many exist in the db

            '
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'','')) '
            + --only if the count matches (all the tables exist in the requested db)
then run the script

            '
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
                Insert  #ApInvoicePay
                            ( DatabaseName
                            , TrnValue
                            , PaymentReference
                            , Supplier
                            , Invoice
                            )
                            Select
                                [DatabaseName] = @DBCode
                              , TrnValue
                              , PaymentReference
                              , Supplier
                              , Invoice
                            From
                                ApInvoicePay;
            End
    End';
        Declare @SQL4 Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end'
            + --Only query DBs beginning SysProCompany

            '
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN'
            + --only companies selected in main run, or if companies selected then
all

            '
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT'
            + --count number of tables requested (number of commas plus one)

            '
```

Author: Johnson, Chris

```
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')'
            + --Count of the tables requested how many exist in the db

            '
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'','')) '
            + --only if the count matches (all the tables exist in the requested db)
then run the script

            '
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
                Insert  #PorMasterHdr
                ( DatabaseName
                , OrderEntryDate
                , PurchaseOrder
                )
                Select
                    [DatabaseName] = @DBCode
                  , OrderEntryDate
                  , PurchaseOrder
                From
                    PorMasterHdr;
            End
    End';
        Declare @SQL5 Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end'
            + --Only query DBs beginning SysProCompany

            '
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN'
            + --only companies selected in main run, or if companies selected then
all

            '
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT'
            + --count number of tables requested (number of commas plus one)

            '
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')'
            + --Count of the tables requested how many exist in the db

            '
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'','')) '
            + --only if the count matches (all the tables exist in the requested db)
then run the script
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-PurchaseOrdersInvoices

```
            '
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
                Insert  #ApSupplier
                ( DatabaseName
                , Supplier
                , SupplierName
                )
                Select
                    [DatabaseName] = @DBCode
                  , Supplier
                  , SupplierName
                From
                    ApSupplier;
            End
    End';

--Enable this function to check script changes (try to run script directly against
db manually)

        /*Print @SQL1;
        Print @SQL2;
        Print @SQL3;
        Print @SQL4;
        Print @SQL5;*/

--execute script against each db, populating the base tables

        Exec [Process].[ExecForEachDB] @cmd = @SQL1;
        Exec [Process].[ExecForEachDB] @cmd = @SQL2;
        Exec [Process].[ExecForEachDB] @cmd = @SQL3;
        Exec [Process].[ExecForEachDB] @cmd = @SQL4;
        Exec [Process].[ExecForEachDB] @cmd = @SQL5;

--define the results you want to return

        Create Table [#Results]
            (
              [DatabaseName] Varchar(150) Collate Latin1_General_BIN
            , [TrnYear] Int
            , [TrnMonth] Int
            , [Supplier] Varchar(35) Collate Latin1_General_BIN
            , [SupplierName] Varchar(150) Collate Latin1_General_BIN
            , [Invoice] Varchar(35) Collate Latin1_General_BIN
            , [InvoiceDate] DateTime2
            , [Reference] Varchar(60) Collate Latin1_General_BIN
            , [PaymentNumber] Varchar(35) Collate Latin1_General_BIN
            , [TrnValue] Numeric(18 , 3)
            , [PaymentReference] Varchar(35) Collate Latin1_General_BIN
            , [PurchaseOrder] Varchar(35) Collate Latin1_General_BIN
            , [OrderEntryDate] DateTime2
            );

--Placeholder to create indexes as required
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-PurchaseOrdersInvoices

```
--script to combine base data and insert into results table

        Insert  [#Results]
                ( [DatabaseName]
                , [TrnYear]
                , [TrnMonth]
                , [Supplier]
                , [SupplierName]
                , [Invoice]
                , [InvoiceDate]
                , [Reference]
                , [PaymentNumber]
                , [TrnValue]
                , [PaymentReference]
                , [PurchaseOrder]
                , [OrderEntryDate]
                )
                Select  [CP].[DatabaseName]
                      , [CP].[CbTrnYear]
                      , [CP].[TrnMonth]
                      , [CP].[Supplier]
                      , [SP].[SupplierName]
                      , [CP].[Invoice]
                      , [CP].[InvoiceDate]
                      , [CP].[Reference]
                      , [CP].[PaymentNumber]
                      , [AIP].[TrnValue]
                      , [AIP].[PaymentReference]
                      , [PMO].[PurchaseOrder]
                      , [PHR].[OrderEntryDate]
                From    [#CshApPayments] [CP]
                        Left Join [#ApInvoice] [AP]
                            On [AP].[DatabaseName] = [CP].[DatabaseName]
                               And [AP].[Supplier] = [CP].[Supplier]
                               And [AP].[Invoice] = [CP].[Invoice]
                               And [AP].[PaymentNumber] = [CP].[PaymentNumber]
                        Left Join [#ApInvoicePay] [AIP]
                            On [AIP].[DatabaseName] = [AP].[DatabaseName]
                               And [AIP].[Supplier] = [AP].[Supplier]
                               And [AIP].[Invoice] = [AP].[Invoice]
                        Left Join [BlackBox].[Lookups].[PurchaseOrderInvoiceMapping]
[PMO]
                            On [PMO].[Invoice] = [CP].[Invoice] Collate Latin1_-
General_BIN
                               And [PMO].[Company] = [CP].[DatabaseName]
                        Left Join [#PorMasterHdr] [PHR]
                            On [PHR].[DatabaseName] = [AIP].[DatabaseName]
                               And [PHR].[PurchaseOrder] = [PMO].[PurchaseOrder]
                        Left Join [#ApSupplier] [SP]
                            On [SP].[Supplier] = [CP].[Supplier]
                Group By [CP].[DatabaseName]
                      , [CP].[CbTrnYear] --group added as there are multiple GRNs on
PMO

                      , [CP].[TrnMonth]
                      , [CP].[Supplier]
                      , [SP].[SupplierName]
                      , [CP].[Invoice]
```

Author: Johnson, Chris

```
                      , [CP].[InvoiceDate]
                      , [CP].[Reference]
                      , [CP].[PaymentNumber]
                      , [AIP].[TrnValue]
                      , [AIP].[PaymentReference]
                      , [PMO].[PurchaseOrder]
                      , [PHR].[OrderEntryDate];

--return results

        Select  [Company] = [DatabaseName]
              , [TrnYear]
              , [TrnMonth]
              , [Supplier]
              , [SupplierName]
              , [Invoice]
              , [InvoiceDate] = Cast([InvoiceDate] As Date)
              , [Reference]
              , [PaymentNumber]
              , [TrnValue] = Coalesce([TrnValue] , 0)
              , [PaymentReference] --= Case When IsNumeric(Coalesce([Payment-
Reference] ,

              --                                              'I')) = 0

              --                            Then [PaymentReference]

              --                            When IsNumeric([PaymentReference]) = 1

              --                            Then Convert(Varchar(25) ,
Convert(Numeric(20) , [PaymentReference]))

              --                            Else [PaymentReference]

              --                       End

              , [PurchaseOrder] --= Case When IsNumeric(Coalesce([PurchaseOrder] ,

                                  --                            'I')) = 0

                                  --     Then [PurchaseOrder]

                                  --     When IsNumeric([PurchaseOrder]) = 1

                                  --     Then Convert(Varchar(25) ,
Convert(Numeric(20) , [PurchaseOrder]))

                                  --     Else [PurchaseOrder]

                                  --End

              , [OrderEntryDate] = Cast([OrderEntryDate] As Date)
              , [Status] = Case When [PurchaseOrder] Is Null
                                Then 'No purchase order'
                                When [OrderEntryDate] Is Null
                                Then 'Unknown entry date'
                                When [OrderEntryDate] >= [InvoiceDate]
                                Then 'Purchase Order raised post receipt of invoice'
```

Author: Johnson, Chris

```
                                Else 'Purchase Order raised on time'
                            End
        From    [#Results];


    End;


GO
EXEC sp_addextendedproperty N'MS_Description', N'purchase order details with
invoices', 'SCHEMA', N'Report', 'PROCEDURE', N'UspResults_PurchaseOrdersInvoices',
NULL, NULL
GO
```

Uses

[Lookups].[PurchaseOrderInvoiceMapping]
[Process].[ExecForEachDB]
[Process].[UspInsert_RedTagLogs]
[Report]

[Report].[UspResults_PurchaseOrdersOpen]

MS_Description

purchase order details for open purchase orders

Parameters

Name	Data Type	Max Length (Bytes)
@Company	varchar(max)	max
@RedTagType	char	1
@RedTagUse	varchar(500)	500

SQL Script

```
CREATE Procedure [Report].[UspResults_PurchaseOrdersOpen]
    (
      @Company Varchar(Max)
    , @RedTagType Char(1)
    , @RedTagUse Varchar(500)
    )
As
    Begin
/*
Template designed by Chris Johnson, Prometic Group September 2015
Stored procedure set out to query multiple databases with the same information and
return it in a collated format
Returns details of all open (non cancelled & non fulfilled) PO's
--Exec [Report].[UspResults_PurchaseOrdersOpen]  10
*/
        If IsNumeric(@Company) = 0
            Begin
                Select  @Company = Upper(@Company);
            End;

--remove nocount on to speed up query

        Set NoCount Off;

--Red tag

        Declare @RedTagDB Varchar(255)= Db_Name();
        Exec [Process].[UspInsert_RedTagLogs] @StoredProcDb = 'BlackBox' ,
            @StoredProcSchema = 'Report' ,
            @StoredProcName = 'UspResults_PurchaseOrdersOpen' ,
            @UsedByType = @RedTagType , @UsedByName = @RedTagUse ,
            @UsedByDb = @RedTagDB;
--list the tables that are to be pulled back from each DB - if they are not found
the script will not be run against that db

        Declare @ListOfTables Varchar(Max) = 'PorMasterHdr,PorMasterDetail,Ap-
Supplier,PorMasterDetail+';
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-PurchaseOrdersOpen

```
--create temporary tables to be pulled from different databases, including a column
to id

        Create Table [#PorMasterHdr]
            (
              [DatabaseName] Varchar(150) Collate Latin1_General_BIN
            , [PurchaseOrder] Varchar(35) Collate Latin1_General_BIN
            , [Buyer] Varchar(35) Collate Latin1_General_BIN
            , [Supplier] Varchar(35) Collate Latin1_General_BIN
            , [OrderStatus] Varchar(35) Collate Latin1_General_BIN
            );
        Create Table [#PorMasterDetail]
            (
              [DatabaseName] Varchar(150) Collate Latin1_General_BIN
            , [PurchaseOrder] Varchar(35) Collate Latin1_General_BIN
            , [Line] Varchar(15) Collate Latin1_General_BIN
            , [StockCode] Varchar(35) Collate Latin1_General_BIN
            , [StockDes] Varchar(150) Collate Latin1_General_BIN
            , [SupCatalogue] Varchar(50) Collate Latin1_General_BIN
            , [OrderQty] Numeric(20 , 7)
            , [ReceivedQty] Numeric(20 , 7)
            , [MPrice] Numeric(20 , 3)
            , [OrderUom] Varchar(10) Collate Latin1_General_BIN
            , [Warehouse] Varchar(35) Collate Latin1_General_BIN
            , [LatestDueDate] DateTime2
            , [CompleteFlag] Char(5)
            , [MForeignPrice] Numeric(20 , 3)
            , [MGlCode] Varchar(35) Collate Latin1_General_BIN
            );
        Create Table [#ApSupplier]
            (
              [DatabaseName] Varchar(150) Collate Latin1_General_BIN
            , [Supplier] Varchar(35) Collate Latin1_General_BIN
            , [SupplierName] Varchar(150) Collate Latin1_General_BIN
            );
        Create Table [#PorMasterDetailPlus]
            (
              [DatabaseName] Varchar(150) Collate Latin1_General_BIN
            , [PurchaseOrder] Varchar(35) Collate Latin1_General_BIN
            , [Line] Varchar(15) Collate Latin1_General_BIN
            , [Confirmed] Varchar(35) Collate Latin1_General_BIN
            );

--create script to pull data from each db into the tables

        Declare @SQL1 Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end'
            + --Only query DBs beginning SysProCompany

            '
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN'
            + --only companies selected in main run, or if companies selected then
all
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-PurchaseOrdersOpen

```
            '
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT'
            + --count number of tables requested (number of commas plus one)

            '
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')'
            + --Count of the tables requested how many exist in the db

            '
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'','')) '
            + --only if the count matches (all the tables exist in the requested db)
then run the script

            '
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
                Insert [#PorMasterHdr]
                        ( [DatabaseName]
                        , [PurchaseOrder]
                        , [Buyer]
                        , [Supplier]
                        , [OrderStatus]
                        )
                SELECT [DatabaseName]=@DBCode
                     , [pmh].[PurchaseOrder]
                     , [pmh].[Buyer]
                     , [pmh].[Supplier]
                     , [OrderStatus]
                FROM [PorMasterHdr] [pmh]
            End
    End';
        Declare @SQL2 Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end'
            + --Only query DBs beginning SysProCompany

            '
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN'
            + --only companies selected in main run, or if companies selected then
all

            '
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT'
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-PurchaseOrdersOpen

```
            + --count number of tables requested (number of commas plus one)

            '
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')'
            + --Count of the tables requested how many exist in the db

            '
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'','')) '
            + --only if the count matches (all the tables exist in the requested db)
then run the script

            '
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
                        Insert [#PorMasterDetail]
                  ( [DatabaseName]
                  , [PurchaseOrder]
                  , [Line]
                  , [StockCode]
                  , [StockDes]
                  , [SupCatalogue]
                  , [OrderQty]
                  , [ReceivedQty]
                  , MPrice
                  , [OrderUom]
                  , [Warehouse]
                  , [LatestDueDate]
                  , [CompleteFlag]
                  , MForeignPrice
                  ,[MGlCode]
                  )
            SELECT [DatabaseName] = @DBCode
                 , [pmd].[PurchaseOrder]
                 , [pmd].[Line]
                 , [pmd].[MStockCode]
                 , [pmd].[MStockDes]
                 , [pmd].[MSupCatalogue]
                 , [pmd].[MOrderQty]
                 , [pmd].[MReceivedQty]
                 , MPrice
                 , [pmd].[MOrderUom]
                 , [pmd].[MWarehouse]
                 , [pmd].[MLatestDueDate]
                 , [pmd].[MCompleteFlag]
                 , [MForeignPrice]
                 ,[MGlCode]
            From [PorMasterDetail] As [pmd]
            End
    End';
        Declare @SQL3 Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end'
            + --Only query DBs beginning SysProCompany
```

Author: Johnson, Chris

```
            '
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN'
            + --only companies selected in main run, or if companies selected then
all

            '
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT'
            + --count number of tables requested (number of commas plus one)

            '
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')'
            + --Count of the tables requested how many exist in the db

            '
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'','')) '
            + --only if the count matches (all the tables exist in the requested db)
then run the script

            '
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
                        Insert [#ApSupplier]
                ( [DatabaseName]
                , [Supplier]
                , [SupplierName]
                )
        SELECT [DatabaseName]=@DBCode
             , [as].[Supplier]
             , [as].[SupplierName] FROM [ApSupplier] As [as]
            End
    End';
        Declare @SQL4 Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end'
            + --Only query DBs beginning SysProCompany

            '
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN'
            + --only companies selected in main run, or if companies selected then
all

            '
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables + '''
                    , @RequiredCountOfTables INT
```

Author: Johnson, Chris

```
                    , @ActualCountOfTables INT'
            + --count number of tables requested (number of commas plus one)

            '
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')'
            + --Count of the tables requested how many exist in the db

            '
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'','')) '
            + --only if the count matches (all the tables exist in the requested db)
then run the script

            '
            print @ActualCountOfTables
            print @RequiredCountOfTables
            If @ActualCountOfTables=@RequiredCountOfTables

            BEGIN
            Insert [#PorMasterDetailPlus]
                    ( [DatabaseName]
                    , [PurchaseOrder]
                    , [Line]
                    , [Confirmed]
                    )
            SELECT [DatabaseName]=@DBCode
                 , [pmdp].[PurchaseOrder]
                 , [pmdp].[Line]
                 , [pmdp].[Confirmed]
            From [PorMasterDetail+] As [pmdp]
            end
    End';

--Enable this function to check script changes (try to run script directly against
db manually)

        --Print @SQL1

        --Print @SQL2

        --Print @SQL3

        --Print @SQL4


--execute script against each db, populating the base tables

        Exec [Process].[ExecForEachDB] @cmd = @SQL1;
        Exec [Process].[ExecForEachDB] @cmd = @SQL2;
        Exec [Process].[ExecForEachDB] @cmd = @SQL3;
        Exec [Process].[ExecForEachDB] @cmd = @SQL4;

--define the results you want to return

        Create Table [#Results]
            (
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-PurchaseOrdersOpen

```
              [Company] Varchar(250) Collate Latin1_General_BIN
            , [PurchaseOrder] Varchar(35) Collate Latin1_General_BIN
            , [Line] Varchar(15) Collate Latin1_General_BIN
            , [Supplier] Varchar(35) Collate Latin1_General_BIN
            , [SupplierName] Varchar(150) Collate Latin1_General_BIN
            , [Buyer] Varchar(35) Collate Latin1_General_BIN
            , [StockCode] Varchar(35) Collate Latin1_General_BIN
            , [StockDescription] Varchar(150) Collate Latin1_General_BIN
            , [SupCatalogue] Varchar(50) Collate Latin1_General_BIN
            , [OrderQty] Numeric(20 , 7)
            , [ReceivedQty] Numeric(20 , 7)
            , [OrderUom] Varchar(10) Collate Latin1_General_BIN
            , [Warehouse] Varchar(35) Collate Latin1_General_BIN
            , [LatestDueDate] Date
            , [Confirmed] Varchar(35) Collate Latin1_General_BIN
            , [OrderStatusDescription] Varchar(150) Collate Latin1_General_BIN
            , [MPrice] Numeric(20 , 2)
            , [MForeignPrice] Numeric(20 , 2)
            , [GLCode] Varchar(35) Collate Latin1_General_BIN
            );

--Placeholder to create indexes as required


--script to combine base data and insert into results table


--return results

        Insert  [#Results]
                ( [Company]
                , [PurchaseOrder]
                , [Line]
                , [Supplier]
                , [SupplierName]
                , [Buyer]
                , [StockCode]
                , [StockDescription]
                , [SupCatalogue]
                , [OrderQty]
                , [ReceivedQty]
                , [OrderUom]
                , [Warehouse]
                , [LatestDueDate]
                , [Confirmed]
                , [OrderStatusDescription]
                , [MPrice]
                , [MForeignPrice]
                , [GLCode]
                )
                Select  [Company] = [PH].[DatabaseName]
                      , [PH].[PurchaseOrder]
                      , [PD].[Line]
                      , [APS].[Supplier]
                      , [APS].[SupplierName]
                      , [PH].[Buyer]
                      , [StockCode] = [PD].[StockCode]
```

Author: Johnson, Chris

```
                      , [StockDescription] = [PD].[StockDes]
                      , [SupCatalogue] = [PD].[SupCatalogue]
                      , [OrderQty] = [PD].[OrderQty]
                      , [ReceivedQty] = [PD].[ReceivedQty]
                      , [OrderUom] = [PD].[OrderUom]
                      , [Warehouse] = [PD].[Warehouse]
                      , [LatestDueDate] = [PD].[LatestDueDate]
                      , [Confirmed] = [PMp].[Confirmed]
                      , [pos].[OrderStatusDescription]
                      , [PD].[MPrice]
                      , [PD].[MForeignPrice]
                      , [GLCode] = [PD].[MGlCode]
                From    [#PorMasterHdr] [PH]
                        Inner Join [#PorMasterDetail] [PD] On [PH].[PurchaseOrder] =
[PD].[PurchaseOrder]
                                                              And [PD].[Database-
Name] = [PH].[DatabaseName]
                        Inner Join [#ApSupplier] [APS] On [PH].[Supplier] =
[APS].[Supplier]
                                                           And [APS].[DatabaseName] =
[PD].[DatabaseName]
                        Left Outer Join [#PorMasterDetailPlus] [PMp] With ( NoLock )
On [PD].[PurchaseOrder] = [PMp].[PurchaseOrder]
                                                              And [PMp].[Database-
Name] = [PD].[DatabaseName]
                                                              And [PD].[Line] =
[PMp].[Line]
                        Left Join [Lookups].[PurchaseOrderStatus] As [pos] On
[APS].[DatabaseName] = [pos].[Company] Collate Latin1_General_BIN
                                                              And [PH].[OrderStatus]
= [pos].[OrderStatusCode] Collate Latin1_General_BIN
                Where   [PH].[OrderStatus] <> '*'
                        And [PD].[OrderQty] > [PD].[ReceivedQty]
                        And ( [PD].[CompleteFlag] <> 'Y' );

        Select  [cn].[CompanyName]
              , [r].[PurchaseOrder]
              , [r].[Line]
              , [r].[Supplier]
              , [r].[SupplierName]
              , [r].[Buyer]
              , [r].[StockCode]
              , [r].[StockDescription]
              , [r].[SupCatalogue]
              , [r].[OrderQty]
              , [r].[ReceivedQty]
              , [r].[OrderUom]
              , [r].[Warehouse]
              , [LatestDueDate] = Cast([r].[LatestDueDate] As Date)
              , [r].[Confirmed]
              , [r].[OrderStatusDescription]
              , [Price] = [r].[MPrice]
              , [ForeignPrice] = [r].[MForeignPrice]
              , [r].[GLCode]
              , [GLCodeDescription] = [GM].[Description]
        From    [#Results] As [r]
                Left Join [Lookups].[CompanyNames] As [cn] On [cn].[Company] =
[r].[Company] Collate Latin1_General_BIN
                Left Join [SysproCompany40].[dbo].[GenMaster] As [GM] On [GM].[Gl-
Code] = [r].[GLCode]
```

Author: Johnson, Chris

```
                                                    And [GM].[Company] =
[r].[Company];


    End;


GO
EXEC sp_addextendedproperty N'MS_Description', N'purchase order details for open
purchase orders', 'SCHEMA', N'Report', 'PROCEDURE', N'UspResults_PurchaseOrders-
Open', NULL, NULL
GO
```

Uses

[Lookups].[CompanyNames]
[Lookups].[PurchaseOrderStatus]
[Process].[ExecForEachDB]
[Process].[UspInsert_RedTagLogs]
[Report]

[Report].[UspResults_PurchaseOrderWithHistory]

MS_Description

purchase order details with purchase order changes

Parameters

Name	Data Type	Max Length (Bytes)
@Company	varchar(max)	max
@RedTagType	char	1
@RedTagUse	varchar(500)	500

SQL Script

```
CREATE Proc [Report].[UspResults_PurchaseOrderWithHistory]
    (
      @Company Varchar(Max)
    , @RedTagType Char(1)
    , @RedTagUse Varchar(500)
    )
As
    Begin
/*
Template designed by Chris Johnson, Prometic Group September 2015
Stored procedure set out to query multiple databases with the same information and
return it in a collated format
*/
        If IsNumeric(@Company) = 0
            Begin
                Select  @Company = Upper(@Company);
            End;

--remove nocount on to speed up query

        Set NoCount On;

--Red tag

        Declare @RedTagDB Varchar(255)= Db_Name();
        Exec [Process].[UspInsert_RedTagLogs] @StoredProcDb = 'BlackBox' ,
            @StoredProcSchema = 'Report' ,
            @StoredProcName = 'UspResults_PurchaseOrderWithHistory' ,
            @UsedByType = @RedTagType , @UsedByName = @RedTagUse ,
            @UsedByDb = @RedTagDB;

--list the tables that are to be pulled back from each DB - if they are not found
the script will not be run against that db

        Declare @ListOfTables Varchar(Max) = 'AssetDepreciation,TblApTerms';

--create temporary tables to be pulled from different databases, including a column
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-PurchaseOrderWithHistory

```
to id

        Create Table [#PorMasterDetail]
            (
              [DatabaseName] Varchar(150)           collate latin1_general_bin
            , [PurchaseOrder] Varchar(20)           collate latin1_general_bin
            , [Line] Int
            , [LineType] Int
            , [MStockCode] Varchar(30)               collate latin1_general_bin
            , [MStockDes] Varchar(50)               collate latin1_general_bin
            , [MWarehouse] Varchar(10)               collate latin1_general_bin
            , [MOrderQty] Numeric(20 , 6)
            , [MReceivedQty] Numeric(20 , 6)
            , [MCompleteFlag] Char(1)               collate latin1_general_bin
            , [MOrderUom] Varchar(10)               collate latin1_general_bin
            , [MLatestDueDate] Date
            , [MOrigDueDate] Date
            , [MLastReceiptDat] Date
            , [MPrice] Numeric(20 , 2)
            , [MForeignPrice] Numeric(20 , 2)
            );
        Create Table [#PorHistReceipt]
            (
              [DatabaseName] Varchar(150)           collate latin1_general_bin
            , [PurchaseOrder] Varchar(20)           collate latin1_general_bin
            , [PurchaseOrderLin] Int
            , [DateReceived] Date
            , [QtyReceived] Numeric(20 , 6)
            , [PriceReceived] Numeric(20 , 2)
            , [RejectCode] Varchar(10)               collate latin1_general_bin
            , [Reference] Varchar(30)               collate latin1_general_bin
            );
        Create Table [#PorMasterHdr]
            (
              [DatabaseName] Varchar(150)           collate latin1_general_bin
            , [PurchaseOrder] Varchar(20)           collate latin1_general_bin
            , [OrderStatus] Char(1)                   collate latin1_general_bin
            , [Supplier] Varchar(15)                collate latin1_general_bin
            );
        Create Table [#InvMaster]
            (
              [DatabaseName] Varchar(150)           collate latin1_general_bin
            , [StockCode] Varchar(30)               collate latin1_general_bin
            , [Description] Varchar(50)               collate latin1_general_bin
            );
        Create Table [#ApSupplier]
            (
              [DatabaseName] Varchar(150)           collate latin1_general_bin
            , [Supplier] Varchar(15)                collate latin1_general_bin
            , [SupplierName] Varchar(50)            collate latin1_general_bin
            );
        Create Table [#GrnDetails]
            (
              [DatabaseName] Varchar(150)           collate latin1_general_bin
            , [Grn] Varchar(20)                       collate latin1_general_bin
            , [DebitRecGlCode] Varchar(35)            collate latin1_general_bin
            );
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-PurchaseOrderWithHistory

```
--create script to pull data from each db into the tables

        Declare @SQL Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables
            + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'',''))
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
                Insert [#PorMasterDetail] ( [DatabaseName], [PurchaseOrder], [Line],
[LineType], [MStockCode], [MStockDes], [MWarehouse], [MOrderQty], [MReceivedQty],
[MCompleteFlag], [MOrderUom], [MLatestDueDate], [MOrigDueDate], [MLastReceiptDat],
[MPrice], [MForeignPrice])
                SELECT [DatabaseName]=@DBCode
                 , [PMD].[PurchaseOrder]
                 , [PMD].[Line]
                 , [PMD].[LineType]
                 , [PMD].[MStockCode]
                 , [PMD].[MStockDes]
                 , [PMD].[MWarehouse]
                 , [PMD].[MOrderQty]
                 , [PMD].[MReceivedQty]
                 , [PMD].[MCompleteFlag]
                 , [PMD].[MOrderUom]
                 , [PMD].[MLatestDueDate]
                 , [PMD].[MOrigDueDate]
                 , [PMD].[MLastReceiptDat]
                 , [PMD].[MPrice]
                 , [PMD].[MForeignPrice] FROM [PorMasterDetail] As [PMD]
                Insert [#PorHistReceipt] ( [DatabaseName], [PurchaseOrder],
[PurchaseOrderLin], [DateReceived], [QtyReceived], [PriceReceived], [RejectCode],
[Reference])
                SELECT [DatabaseName]=@DBCode
                     , [PHR].[PurchaseOrder]
                     , [PHR].[PurchaseOrderLin]
                     , [PHR].[DateReceived]
                     , [PHR].[QtyReceived]
                     , [PHR].[PriceReceived]
                     , [PHR].[RejectCode]
                     , [PHR].[Reference] FROM [PorHistReceipt] As [PHR]
                Insert [#PorMasterHdr] ( [DatabaseName], [PurchaseOrder], [Order-
Status], [Supplier])
                SELECT [DatabaseName]=@DBCode
```

Author: Johnson, Chris

```
                    , [PMH].[PurchaseOrder]
                    , [PMH].[OrderStatus]
                    , [PMH].[Supplier] FROM [PorMasterHdr] As [PMH]
               Insert [#InvMaster] ( [DatabaseName], [StockCode], [Description])
               SELECT [DatabaseName]=@DBCode
                    , [IM].[StockCode]
                    , [IM].[Description] FROM [InvMaster] As [IM]
               Insert [#ApSupplier] ( [DatabaseName], [Supplier], [SupplierName])
               SELECT [DatabaseName]=@DBCode
                    , [AS].[Supplier]
                    , [AS].[SupplierName] FROM [ApSupplier] As [AS]
               Insert [#GrnDetails] ( [DatabaseName], [Grn], [DebitRecGlCode])
               Select  [DatabaseName] = @DBCode
                     , [GD].[Grn]
                     , [GD].[DebitRecGlCode]
               From    [GrnDetails] As [GD];
          End
    End';

--Enable this function to check script changes (try to run script directly against
db manually)

--Print @SQL


--execute script against each db, populating the base tables

        Exec [Process].[ExecForEachDB] @cmd = @SQL;

--define the results you want to return

        Create Table [#Results]
            (
              [DatabaseName] Varchar(150)            collate latin1_general_bin
            , [CompanyName] Varchar(200)            collate latin1_general_bin
            , [Supplier] Varchar(15)                collate latin1_general_bin
            , [SupplierName] Varchar(50)            collate latin1_general_bin
            , [PurchaseOrder] Varchar(30)           collate latin1_general_bin
            , [Line] Int
            , [LineType] Varchar(200)               collate latin1_general_bin
            , [StockCode] Varchar(30)               collate latin1_general_bin
            , [StockDescription] Varchar(50)        collate latin1_general_bin
            , [Warehouse] Varchar(10)               collate latin1_general_bin
            , [OrderQty] Numeric(20 , 6)
            , [ReceivedQty] Numeric(20 , 6)
            , [QtyOutstanding] Numeric(20 , 6)
            , [MOrderUom] Varchar(10)               collate latin1_general_bin
            , [LatestDueDate] Date
            , [OrigDueDate] Date
            , [LastReceiptDate] Date
            , [Price] Numeric(20 , 2)
            , [ForeignPrice] Numeric(20 , 2)
            , [DateReceived] Date
            , [QtyReceived] Numeric(20 , 6)
            , [PriceReceived] Numeric(20 , 2)
            , [RejectCode] Varchar(10)              collate latin1_general_bin
            , [Reference] Varchar(30)               collate latin1_general_bin
```

Author: Johnson, Chris

```
            , [CompleteFlag] Char(1)                collate latin1_general_bin
            , [OrderStatus] Varchar(150)            collate latin1_general_bin
            , [DebitRecGlCode] Varchar(35)            collate latin1_general_bin
            );

--Placeholder to create indexes as required


--script to combine base data and insert into results table

        Insert  [#Results]
                ( [DatabaseName]
                , [CompanyName]
                , [Supplier]
                , [SupplierName]
                , [PurchaseOrder]
                , [Line]
                , [LineType]
                , [StockCode]
                , [StockDescription]
                , [Warehouse]
                , [OrderQty]
                , [ReceivedQty]
                , [QtyOutstanding]
                , [MOrderUom]
                , [LatestDueDate]
                , [OrigDueDate]
                , [LastReceiptDate]
                , [Price]
                , [ForeignPrice]
                , [DateReceived]
                , [QtyReceived]
                , [PriceReceived]
                , [RejectCode]
                , [Reference]
                , [CompleteFlag]
                , [OrderStatus]
                , [DebitRecGlCode]
                )
                Select  [PMD].[DatabaseName]
                      , [CN].[CompanyName]
                      , [PMH].[Supplier]
                      , [AS].[SupplierName]
                      , [PMD].[PurchaseOrder]
                      , [PMD].[Line]
                      , [LineType] = [PLT].[PorLineTypeDesc]
                      , [StockCode] = [PMD].[MStockCode]
                      , [StockDescription] = Coalesce([IM].[Description] ,
                                                      [PMD].[MStockDes])
                      , [Warehouse] = [PMD].[MWarehouse]
                      , [OrderQty] = [PMD].[MOrderQty]
                      , [ReceivedQty] = [PMD].[MReceivedQty]
                      , [QtyOutstanding] = Case When [PMD].[MCompleteFlag] = 'Y'
                                                Then Convert(Numeric(20 , 6) , 0)
                                                Else [PMD].[MOrderQty]
                                                     - [PMD].[MReceivedQty]
                                           End
```

Author: Johnson, Chris

```
                        , [PMD].[MOrderUom]
                        , [LatestDueDate] = Convert(Date , [PMD].[MLatestDueDate])
                        , [OrigDueDate] = Convert(Date , [PMD].[MOrigDueDate])
                        , [LastReceiptDate] = Convert(Date , [PMD].[MLastReceiptDat])
                        , [Price] = [PMD].[MPrice]
                        , [ForeignPrice] = [PMD].[MForeignPrice]
                        , [DateReceived] = Convert(Date , [PHR].[DateReceived])
                        , [PHR].[QtyReceived]
                        , [PHR].[PriceReceived]
                        , [RejectCode] = Case When [PHR].[RejectCode] = ''
                                              Then Null
                                              Else [PHR].[RejectCode]
                                         End
                        , [PHR].[Reference]
                        , [CompleteFlag] = Case When [PMD].[MCompleteFlag] = ''
                                                Then Null
                                                Else [PMD].[MCompleteFlag]
                                           End
                        , [OrderStatus] = [POS].[OrderStatusDescription]
                        , [GD].[DebitRecGlCode]
                  From    [#PorMasterDetail] As [PMD]
                          Left Join [#PorHistReceipt] As [PHR] On [PHR].[Purchase-
Order] = [PMD].[PurchaseOrder]
                                                                  And [PMD].[Line] =
[PHR].[PurchaseOrderLin]
                                                                  And [PHR].[Database-
Name] = [PMD].[DatabaseName]
                          Left Join [#PorMasterHdr] As [PMH] On [PMH].[PurchaseOrder]
= [PHR].[PurchaseOrder]
                                                                  And [PMH].[Database-
Name] = [PHR].[DatabaseName]
                          Left Join [BlackBox].[Lookups].[PurchaseOrderStatus]
                          As [POS] On [POS].[OrderStatusCode] = [PMH].[OrderStatus]
                                      And [POS].[Company] = [PMH].[DatabaseName]
                          Left Join [#InvMaster] As [IM] On [IM].[StockCode] =
[PMD].[MStockCode]
                                                            And [IM].[DatabaseName] =
[PMD].[DatabaseName]
                          Left Join [#ApSupplier] As [AS] On [AS].[Supplier] =
[PMH].[Supplier]
                                                             And [AS].[DatabaseName] =
[PMH].[DatabaseName]
                          Left Join [BlackBox].[Lookups].[PorLineType] As [PLT] On
[PLT].[PorLineType] = [PMD].[LineType]
                          Left Join [#GrnDetails] As [GD] On [PHR].[Reference] =
[GD].[Grn]
                                                             And [GD].[DatabaseName] =
[PHR].[DatabaseName]
                          Left Join [BlackBox].[Lookups].[CompanyNames] As [CN] On
[CN].[Company] = [PMD].[DatabaseName];


--return results

        Select  [DatabaseName]
              , [CompanyName]
              , [Supplier]
              , [SupplierName]
              , [PurchaseOrder]
              , [Line]
```

Author: Johnson, Chris

```
                , [LineType]
                , [StockCode]
                , [StockDescription]
                , [Warehouse]
                , [OrderQty]
                , [ReceivedQty]
                , [QtyOutstanding]
                , [MOrderUom]
                , [LatestDueDate]
                , [OrigDueDate]
                , [LastReceiptDate]
                , [Price]
                , [ForeignPrice]
                , [DateReceived]
                , [QtyReceived]
                , [PriceReceived]
                , [RejectCode]
                , [Reference]
                , [CompleteFlag]
                , [OrderStatus]
                , [GlCode] = [DebitRecGlCode]
                , [GLDescription] = [GM].[Description]
        From    [#Results]
                  Left Join [SysproCompany40].[dbo].[GenMaster] As [GM] On [DebitRecGl-
Code] = [GM].[GlCode]
                                                                And [DatabaseName] =
[GM].[Company];

    End;

GO
EXEC sp_addextendedproperty N'MS_Description', N'purchase order details with
purchase order changes', 'SCHEMA', N'Report', 'PROCEDURE', N'UspResults_Purchase-
OrderWithHistory', NULL, NULL
GO
```

Uses

[Lookups].[CompanyNames]
[Lookups].[PorLineType]
[Lookups].[PurchaseOrderStatus]
[Process].[ExecForEachDB]
[Process].[UspInsert_RedTagLogs]
[Report]

Author: Johnson, Chris

[Report].[UspResults_ReceivedLotJob]

Parameters

Name	Data Type	Max Length (Bytes)
@Company	varchar(max)	max
@Lot	varchar(50)	50
@RedTagType	char	1
@RedTagUse	varchar(500)	500

SQL Script

```
CREATE Proc [Report].[UspResults_ReceivedLotJob]
    (
      @Company Varchar(Max)
    , @Lot Varchar(50)
    , @RedTagType Char(1)
    , @RedTagUse Varchar(500)
    )
As
    Begin
        If IsNumeric(@Company) = 0
            Begin
                Select  @Company = Upper(@Company);
            End;

        Set @Lot = Case When IsNumeric(@Lot)=1 Then
Convert(Varchar(50),Convert(Int,@Lot)) Else Upper(@Lot) End
--remove nocount on to speed up query

        Set NoCount On;

--Red tag

        Declare @RedTagDB Varchar(255)= Db_Name();
        Exec [Process].[UspInsert_RedTagLogs] @StoredProcDb = 'BlackBox' ,
            @StoredProcSchema = 'Report' ,
            @StoredProcName = 'UspResults_ReceivedLotJob' ,
            @UsedByType = @RedTagType , @UsedByName = @RedTagUse ,
            @UsedByDb = @RedTagDB;

--list the tables that are to be pulled back from each DB - if they are not found
the script will not be run against that db

        Declare @ListOfTables Varchar(Max) = 'LotTransactions';

--create temporary tables to be pulled from different databases, including a column
to id

        Create Table [#LotTransactions]
            (
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-ReceivedLotJob

```
              [DatabaseName] Varchar(150)
            , [Lot] Varchar(50)
            , [JobPurchOrder] Varchar(20)
            );

--create script to pull data from each db into the tables

        Declare @SQL Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            BEGIN
                Insert [#LotTransactions]
                ( [DatabaseName]
                , [Lot]
                , [JobPurchOrder]
                )
        SELECT [DatabaseName]=@DBCode
             , [LT].[Lot]
             , [LT].[JobPurchOrder]
        From [LotTransactions] [LT]
        where Case When IsNumeric(LT.Lot)=1 Then
Convert(Varchar(50),Convert(Int,LT.Lot)) Else Upper(LT.Lot) End = ''' + @Lot + '''
        and [LT].[TrnType]=''R''
    And [LT].[JobPurchOrder]<>''''
            End
    End';

--execute script against each db, populating the base tables

        Exec [Process].[ExecForEachDB_WithTableCheck] @cmd = @SQL , -- nvarchar(max)

            @SchemaTablesToCheck = @ListOfTables; -- nvarchar(max)



--define the results you want to return


--Placeholder to create indexes as required


--script to combine base data and insert into results table


        Set NoCount Off;
--return results

        Select  [LT].[DatabaseName]
              , [LT].[Lot]
              , [LT].[JobPurchOrder]
        From    [#LotTransactions] [LT];
```

Author: Johnson, Chris

```
    End;

GO
```

Uses

[Process].[ExecForEachDB_WithTableCheck]
[Process].[UspInsert_RedTagLogs]
[Report]

[Report].[UspResults_RedTagDetails]

MS_Description

provides details of red tag logs (logs generated by other SPs)

SQL Script

```
CREATE Proc [Report].[UspResults_RedTagDetails]
/*
Stored procedure created by Chris Johnson 2nd to show all procs marked to be
included in Red Tag review

*/
As --Get list of procs with Red Tag Type included in parameter

    Create Table [#Procs]
        (
          [SchemaName] Varchar(255)            collate latin1_general_bin
        , [ProcedureName] Varchar(255)        collate latin1_general_bin
        , [ParameterDetails] Varchar(Max)    collate latin1_general_bin
        , [DatabaseName] Varchar(500)        collate latin1_general_bin
        , [ExecScript] Varchar(Max)            collate latin1_general_bin
        , [CreateScript] Varchar(Max)        collate latin1_general_bin
        );

    Insert  [#Procs]
            ( [SchemaName]
            , [ProcedureName]
            , [ParameterDetails]
            , [DatabaseName]
            , [ExecScript]
            , [CreateScript]
            )
            Exec [Reports].[UspResults_SearchForProcedures] @DbSearch = '' ,
                @ProcedureSearch = '' , @SchemaSearch = '' ,
                @ParameterSearch = 'RedTagType';


--Provide results from Red Tag

    Select  [P].[DatabaseName]
          , [P].[SchemaName]
          , [P].[ProcedureName]
          , [P].[ParameterDetails]
          , [RTUBT].[UsedByDescription]
          , [CountOfRuns] = Count([RTL].[TagID])
          , [LatestRun] = Max([RTL].[TagDatetime])
          , [DaysSinceLastRun] = Case When Max([RTL].[TagDatetime]) Is Null
                                      Then 'Never run'
                                      Else Cast(DateDiff(Day ,
                                                         Max([RTL].[TagDatetime]) ,
                                                         GetDate()) As Varchar(10))
                                 End
```

Author: Johnson, Chris

```
          , [Rankings] = dense_rank() Over (Order By Case
                                                                When DateDiff(Day ,
                                                                Max([RTL].[Tag-
Datetime]) ,
                                                                GetDate()) < 7
                                                                Then 1000000
                                                                + Count([RTL].[TagID])
                                                                When DateDiff(Day ,
                                                                Max([RTL].[Tag-
Datetime]) ,
                                                                GetDate()) < 30
                                                                Then 900000
                                                                + Count([RTL].[TagID])
                                                                Else Count([RTL].[Tag-
ID])
                                                                End Desc )
    From    [#Procs] As [P]
            Left Join [History].[RedTagLogs] As [RTL] On [P].[DatabaseName] =
[RTL].[StoredProcDb]
                                                        And [P].[ProcedureName] =
[RTL].[StoredProcName]
                                                        And [P].[SchemaName] =
[RTL].[StoredProcSchema]
            Left Join [Lookups].[RedTagsUsedByType] As [RTUBT] On [RTUBT].[UsedBy-
Type] = [RTL].[UsedByType]
    Group By [P].[SchemaName]
          , [P].[ProcedureName]
          , [P].[ParameterDetails]
          , [P].[DatabaseName]
          , [RTUBT].[UsedByDescription]
    Order By [Rankings] Asc;
    Drop Table [#Procs];
GO
EXEC sp_addextendedproperty N'MS_Description', N'provides details of red tag logs
(logs generated by other SPs)', 'SCHEMA', N'Report', 'PROCEDURE', N'UspResults_Red-
TagDetails', NULL, NULL
GO
```

Uses

[History].[RedTagLogs]
[Lookups].[RedTagsUsedByType]
[Reports].[UspResults_SearchForProcedures]
[Report]

Author: Johnson, Chris

[Report].[UspResults_ReportSysRefresh_RedTagLogs]

MS_Description

returns red tag logs for sys refresh times to ensure this is working and the average time a run takes

Parameters

Name	Data Type	Max Length (Bytes)
@DatePart	varchar(500)	500
@StartPeriod	datetime	8
@EndPeriod	datetime	8
@RedTagType	char	1
@RedTagUse	varchar(500)	500

SQL Script

```
CREATE Proc [Report].[UspResults_ReportSysRefresh_RedTagLogs]
    (
      @DatePart Varchar(500)
    , @StartPeriod DateTime
    , @EndPeriod DateTime
    , @RedTagType Char(1)
    , @RedTagUse Varchar(500)
    )
As
    Begin
        Set NoCount On;
        Set @DatePart = Lower(@DatePart);

--Red tag

        Declare @RedTagDB Varchar(255)= Db_Name();
        Exec [Process].[UspInsert_RedTagLogs] @StoredProcDb = 'BlackBox' ,
            @StoredProcSchema = 'Report' ,
            @StoredProcName = 'UspResults_ReportSysRefresh_RedTagLogs' ,
            @UsedByType = @RedTagType , @UsedByName = @RedTagUse ,
            @UsedByDb = @RedTagDB;

--create temporary tables to be pulled from different databases, including a column
to id

        Create Table [#TranRanks]
            (
              [TagID] Int
            , [TagDatetime] DateTime2
            , [StoredProcDb] Varchar(255) Collate Latin1_General_BIN
            , [StoredProcSchema] Varchar(255) Collate Latin1_General_BIN
            , [StoredProcName] Varchar(255) Collate Latin1_General_BIN
            , [UsedByName] Varchar(500) Collate Latin1_General_BIN
            , [UsedByDb] Varchar(255) Collate Latin1_General_BIN
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-ReportSysRefresh_RedTagLogs

```
            , [TranRank] BigInt
            );
        Create Table [#ListOfRuns]
            (
              [StartDate] DateTime
            , [EndDate] DateTime
            , [SecondsToRun] BigInt
            );


--create script to pull data from each db into the tables

        Insert  [#TranRanks]
                ( [TagID]
                , [TagDatetime]
                , [StoredProcDb]
                , [StoredProcSchema]
                , [StoredProcName]
                , [UsedByName]
                , [UsedByDb]
                , [TranRank]
                )
                Select  [RTL].[TagID]
                      , [RTL].[TagDatetime]
                      , [RTL].[StoredProcDb]
                      , [RTL].[StoredProcSchema]
                      , [RTL].[StoredProcName]
                      , [RTL].[UsedByName]
                      , [RTL].[UsedByDb]
                      , [TranRank] = Rank() Over ( Order By [RTL].[TagID] Desc )
                From    [History].[RedTagLogs] [RTL]
                Where   [RTL].[UsedByName] Like 'Development > Report System Refresh
>%'
                        And [RTL].[TagDatetime] Between @StartPeriod
                                                    And     @EndPeriod
                Order By [RTL].[TagID] Desc;

        Insert  [#ListOfRuns]
                ( [StartDate]
                , [EndDate]
                , [SecondsToRun]
                )
                Select  [StartDate] = Convert(DateTime , [T].[TagDatetime])
                      , [EndDate] = Convert(DateTime , [T2].[TagDatetime])
                      , [SecondsToRun] = DateDiff(Second , [T].[TagDatetime] ,
                                                  [T2].[TagDatetime])
                From    [#TranRanks] [T]
                        Left Join [#TranRanks] [T2]
                            On [T2].[TranRank] = [T].[TranRank] - 1
                Where   [T].[UsedByName] Like '%Started%';




--define the results you want to return
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-ReportSysRefresh_RedTagLogs

```
        Create Table [#Results]
            (
              [DatePartTime] DateTime
            , [DatePartName] Varchar(255)
            , [CountOfStarts] BigInt
            , [AvgSecondsToRun] BigInt
            , [MaxSecondsToRun] BigInt
            , [MinSecondsToRun] BigInt
            , [CountOfUnfinishedRuns] BigInt
            );

--Placeholder to create indexes as required


--script to combine base data and insert into results table



        If @DatePart = 'hour'
            Begin
                Insert  [#Results]
                        ( [DatePartTime]
                        , [DatePartName]
                        , [CountOfStarts]
                        , [AvgSecondsToRun]
                        , [MaxSecondsToRun]
                        , [MinSecondsToRun]
                        , [CountOfUnfinishedRuns]
                        )
                        Select  [DatePartTime] = DateAdd(Hour ,
                                                         DateDiff(Hour , 0 ,
                                                              [T].[StartDate]) ,
                                                         0)
                              , [DatePartName] = 'Hour'
                              , [CountOfStarts] = Count(Distinct [T].[StartDate])
                              , [AvgSecondsToRun] = Avg([T].[SecondsToRun])
                              , [MaxSecondsToRun] = Max([T].[SecondsToRun])
                              , [MinSecondsToRun] = Min([T].[SecondsToRun])
                              , [CountOfUnfinishedRuns] = Sum(Case
                                                              When [T].[EndDate] Is
Null
                                                              Then 1
                                                              Else 0
                                                              End)
                        From    [#ListOfRuns] [T]
                        Group By DateAdd(Hour ,
                                         DateDiff(Hour , 0 , [T].[StartDate]) ,
                                         0);
            End;
        If @DatePart = 'day'
            Begin
                Insert  [#Results]
                        ( [DatePartTime]
                        , [DatePartName]
                        , [CountOfStarts]
                        , [AvgSecondsToRun]
                        , [MaxSecondsToRun]
```

Author: Johnson, Chris

```
                        , [MinSecondsToRun]
                        , [CountOfUnfinishedRuns]
                        )
                        Select  [DatePartTime] = DateAdd(Day ,
                                                         DateDiff(Day , 0 ,
                                                              [T].[StartDate]) ,
                                                         0)
                              , [DatePartName] = 'Day'
                              , [CountOfStarts] = Count(Distinct [T].[StartDate])
                              , [AvgSecondsToRun] = Avg([T].[SecondsToRun])
                              , [MaxSecondsToRun] = Max([T].[SecondsToRun])
                              , [MinSecondsToRun] = Min([T].[SecondsToRun])
                              , [CountOfUnfinishedRuns] = Sum(Case
                                                              When [T].[EndDate] Is
Null
                                                              Then 1
                                                              Else 0
                                                              End)
                        From    [#ListOfRuns] [T]
                        Group By DateAdd(Day ,
                                         DateDiff(Day , 0 , [T].[StartDate]) ,
                                         0);
            End;
        If @DatePart = 'week'
            Begin
                Insert  [#Results]
                        ( [DatePartTime]
                        , [DatePartName]
                        , [CountOfStarts]
                        , [AvgSecondsToRun]
                        , [MaxSecondsToRun]
                        , [MinSecondsToRun]
                        , [CountOfUnfinishedRuns]
                        )
                        Select  [DatePartTime] = DateAdd(Week ,
                                                         DateDiff(Week , 0 ,
                                                              [T].[StartDate]) ,
                                                         0)
                              , [DatePartName] = 'Week'
                              , [CountOfStarts] = Count(Distinct [T].[StartDate])
                              , [AvgSecondsToRun] = Avg([T].[SecondsToRun])
                              , [MaxSecondsToRun] = Max([T].[SecondsToRun])
                              , [MinSecondsToRun] = Min([T].[SecondsToRun])
                              , [CountOfUnfinishedRuns] = Sum(Case
                                                              When [T].[EndDate] Is
Null
                                                              Then 1
                                                              Else 0
                                                              End)
                        From    [#ListOfRuns] [T]
                        Group By DateAdd(Week ,
                                         DateDiff(Week , 0 , [T].[StartDate]) ,
                                         0);
            End;
        If @DatePart = 'month'
            Begin
                Insert  [#Results]
```

Author: Johnson, Chris

```
                        ( [DatePartTime]
                        , [DatePartName]
                        , [CountOfStarts]
                        , [AvgSecondsToRun]
                        , [MaxSecondsToRun]
                        , [MinSecondsToRun]
                        , [CountOfUnfinishedRuns]
                        )
                        Select  [DatePartTime] = DateAdd(Month ,
                                                         DateDiff(Month , 0 ,
                                                              [T].[StartDate]) ,
                                                         0)
                              , [DatePartName] = 'Month'
                              , [CountOfStarts] = Count(Distinct [T].[StartDate])
                              , [AvgSecondsToRun] = Avg([T].[SecondsToRun])
                              , [MaxSecondsToRun] = Max([T].[SecondsToRun])
                              , [MinSecondsToRun] = Min([T].[SecondsToRun])
                              , [CountOfUnfinishedRuns] = Sum(Case
                                                              When [T].[EndDate] Is
Null
                                                              Then 1
                                                              Else 0
                                                              End)
                        From    [#ListOfRuns] [T]
                        Group By DateAdd(Month ,
                                         DateDiff(Month , 0 , [T].[StartDate]) ,
                                         0);
            End;
        If @DatePart = 'year'
            Begin
                Insert  [#Results]
                        ( [DatePartTime]
                        , [DatePartName]
                        , [CountOfStarts]
                        , [AvgSecondsToRun]
                        , [MaxSecondsToRun]
                        , [MinSecondsToRun]
                        , [CountOfUnfinishedRuns]
                        )
                        Select  [DatePartTime] = DateAdd(Year ,
                                                         DateDiff(Year , 0 ,
                                                              [T].[StartDate]) ,
                                                         0)
                              , [DatePartName] = 'Year'
                              , [CountOfStarts] = Count(Distinct [T].[StartDate])
                              , [AvgSecondsToRun] = Avg([T].[SecondsToRun])
                              , [MaxSecondsToRun] = Max([T].[SecondsToRun])
                              , [MinSecondsToRun] = Min([T].[SecondsToRun])
                              , [CountOfUnfinishedRuns] = Sum(Case
                                                              When [T].[EndDate] Is
Null
                                                              Then 1
                                                              Else 0
                                                              End)
                        From    [#ListOfRuns] [T]
                        Group By DateAdd(Year ,
                                         DateDiff(Year , 0 , [T].[StartDate]) ,
```

Author: Johnson, Chris

```
                                        0);
            End;


        Set NoCount Off;
--return results

        Select  [DatePartTime]
              , [DatePartName]
              , [CountOfStarts]
              , [AvgSecondsToRun]
              , [MaxSecondsToRun]
              , [MinSecondsToRun]
              , [CountOfUnfinishedRuns]
        From    [#Results]
        Order By [DatePartTime] Desc;

    End;

GO
EXEC sp_addextendedproperty N'MS_Description', N'returns red tag logs for sys
refresh times to ensure this is working and the average time a run takes', 'SCHEMA',
N'Report', 'PROCEDURE', N'UspResults_ReportSysRefresh_RedTagLogs', NULL, NULL
GO
```

Uses

[History].[RedTagLogs]
[Process].[UspInsert_RedTagLogs]
[Report]

Author: Johnson, Chris

[Report].[UspResults_ReqsAfterInvoices]

MS_Description

list of requisitions raised post invoices appearing in system

Parameters

Name	Data Type	Max Length (Bytes)
@Company	varchar(max)	max
@RedTagType	char	1
@RedTagUse	varchar(500)	500

SQL Script

```
CREATE Proc [Report].[UspResults_ReqsAfterInvoices]
    (
      @Company Varchar(Max)
    , @RedTagType Char(1)
    , @RedTagUse Varchar(500)
    )
As
    Begin
/*
Template designed by Chris Johnson, Prometic Group September 2015
Stored procedure set out to query multiple databases with the same information and
return it in a collated format
Proc to return where a requisition was created after an invoice was received
*/
        If IsNumeric(@Company) = 0
            Begin
                Select  @Company = Upper(@Company);
            End;

--remove nocount on to speed up query

        Set NoCount On;

--Red tag

        Declare @RedTagDB Varchar(255)= Db_Name();
        Exec [Process].[UspInsert_RedTagLogs] @StoredProcDb = 'BlackBox' ,
            @StoredProcSchema = 'Report' ,
            @StoredProcName = 'UspResults_ReqsAfterInvoices' ,
            @UsedByType = @RedTagType , @UsedByName = @RedTagUse ,
            @UsedByDb = @RedTagDB;

--list the tables that are to be pulled back from each DB - if they are not found
the script will not be run against that db

        Declare @ListOfTables Varchar(Max) = 'ReqHeader,ReqDetail,PorMasterDetail';
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_Reqs-AfterInvoices

```
--create temporary tables to be pulled from different databases, including a column
to id

        Create Table [#ReqHeader]
            (
              [DatabaseName] Varchar(150)            collate Latin1_General_BIN
            , [Requisition] Varchar(10)                collate Latin1_General_BIN
            , [DateReqnRaised] Date
            );
        Create Table [#ReqDetail]
            (
              [DatabaseName] Varchar(150)            collate Latin1_General_BIN
            , [Requisition] Varchar(10)                collate Latin1_General_BIN
            , [Line] Int
            , [ApprovedDate] Date
            , [StockCode] Varchar(30)                collate Latin1_General_BIN
            , [StockDescription] Varchar(50)        collate Latin1_General_BIN
            );
        Create Table [#PorMasterDetail]
            (
              [DatabaseName] Varchar(150)            collate Latin1_General_BIN
            , [MRequisition] Varchar(10)            collate Latin1_General_BIN
            , [Line] Int
            , [PurchaseOrder] Varchar(20)            collate Latin1_General_BIN
            , [MOrderQty] Numeric(20 , 8)
            , [MPrice] Numeric(20 , 2)
            , [MForeignPrice] Numeric(20 , 2)
            );
        Create Table [#PorMasterHdr]
            (
              [DatabaseName] Varchar(150)            collate Latin1_General_BIN
            , [PurchaseOrder] Varchar(20)            collate Latin1_General_BIN
            , [OrderEntryDate] Date
            );
        Create Table [#GrnDetails]
            (
              [DatabaseName] Varchar(150)            collate Latin1_General_BIN
            , [PurchaseOrder] Varchar(20)            collate Latin1_General_BIN
            , [Grn] Varchar(20)                        collate Latin1_General_BIN
            , [Supplier] Varchar(15)                collate Latin1_General_BIN
            , [Warehouse] Varchar(10)                collate Latin1_General_BIN
            );
        Create Table [#GrnMatching]
            (
              [DatabaseName] Varchar(150)            collate Latin1_General_BIN
            , [Grn] Varchar(20)                        collate Latin1_General_BIN
            , [Supplier] Varchar(15)                collate Latin1_General_BIN
            , [Invoice] Varchar(20)                    collate Latin1_General_BIN
            );
        Create Table [#ApInvoice]
            (
              [DatabaseName] Varchar(150)            collate Latin1_General_BIN
            , [Supplier] Varchar(15)                collate Latin1_General_BIN
            , [Invoice] Varchar(20)                    collate Latin1_General_BIN
            , [InvoiceDate] Date
            , [PostCurrency] Varchar(10)            collate Latin1_General_BIN
            );
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_Reqs-AfterInvoices

```
        Create Table [#ApSupplier]
            (
              [DatabaseName] Varchar(150)            collate Latin1_General_BIN
            , [Supplier] Varchar(15)                collate Latin1_General_BIN
            , [SupplierName] Varchar(50)            collate Latin1_General_BIN
            );

--create script to pull data from each db into the tables

        Declare @SQLReqs Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end'
            + --Only query DBs beginning SysProCompany

            '
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN'
            + --only companies selected in main run, or if companies selected then
all

            '
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT'
            + --count number of tables requested (number of commas plus one)

            '
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')'
            + --Count of the tables requested how many exist in the db

            '
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'','')) '
            + --only if the count matches (all the tables exist in the requested db)
then run the script

            '
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
                Insert [#ReqHeader]
                        ( [DatabaseName]
                        , [Requisition]
                        , [DateReqnRaised]
                        )
                SELECT [DatabaseName]=@DBCode
                     , [RH].[Requisition]
                     , [RH].[DateReqnRaised] FROM [ReqHeader] As [RH]

                Insert [#ReqDetail]
                        ( [DatabaseName]
                        , [Requisition]
                        , [Line]
```

Author: Johnson, Chris

```
                        , [ApprovedDate]
                        , [StockCode]
                        , [StockDescription]
                        )
                SELECT [DatabaseName]=@DBCode
                     , [RD].[Requisition]
                     , [RD].[Line]
                     , [RD].[ApprovedDate]
                     , [RD].[StockCode]
                     , [RD].[StockDescription] FROM [ReqDetail] As [RD]
            End
    End';
        Declare @SQLPorMasters Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end'
            + --Only query DBs beginning SysProCompany

            '
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN'
            + --only companies selected in main run, or if companies selected then
all

            '
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT'
            + --count number of tables requested (number of commas plus one)

            '
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')'
            + --Count of the tables requested how many exist in the db

            '
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'','')) '
            + --only if the count matches (all the tables exist in the requested db)
then run the script

            '
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
                Insert [#PorMasterDetail]
                        ( [DatabaseName]
                        , [MRequisition]
                        , [Line]
                        , [PurchaseOrder]
                        , [MOrderQty]
                        , [MPrice]
                        , [MForeignPrice]
                        )
                SELECT [DatabaseName]=@DBCode
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_Reqs-AfterInvoices

```
                    , [PMD].[MRequisition]
                    , [PMD].[Line]
                    , [PMD].[PurchaseOrder]
                    , [PMD].[MOrderQty]
                    , [PMD].[MPrice]
                    , [PMD].[MForeignPrice] FROM [PorMasterDetail] As [PMD]

                Insert [#PorMasterHdr]
                        ( [DatabaseName]
                        , [PurchaseOrder]
                        , [OrderEntryDate]
                        )
                SELECT [DatabaseName]=@DBCode
                     , [PMH].[PurchaseOrder]
                     , [PMH].[OrderEntryDate] FROM [PorMasterHdr] As [PMH]
            End
    End';
        Declare @SQLGrns Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end'
            + --Only query DBs beginning SysProCompany

            '
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN'
            + --only companies selected in main run, or if companies selected then
all

            '
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT'
            + --count number of tables requested (number of commas plus one)

            '
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')'
            + --Count of the tables requested how many exist in the db

            '
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'','')) '
            + --only if the count matches (all the tables exist in the requested db)
then run the script

            '
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
            Insert [#GrnDetails]
                    ( [DatabaseName]
                    , [PurchaseOrder]
                    , [Grn]
                    , [Supplier]
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_Reqs-AfterInvoices

```
                    , [Warehouse]
                    )
            SELECT [DatabaseName]=@DBCode
                 , [GD].[PurchaseOrder]
                 , [GD].[Grn]
                 , [GD].[Supplier]
                 , [GD].[Warehouse] FROM [GrnDetails] As [GD]

            Insert [#GrnMatching]
                    ( [DatabaseName]
                    , [Grn]
                    , [Supplier]
                    , [Invoice]
                    )
            SELECT [DatabaseName]=@DBCode
                 , [GM].[Grn]
                 , [GM].[Supplier]
                 , [GM].[Invoice] FROM [GrnMatching] As [GM]
            End
    End';
        Declare @SQLAps Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end'
            + --Only query DBs beginning SysProCompany

            '
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN'
            + --only companies selected in main run, or if companies selected then
all

            '
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT'
            + --count number of tables requested (number of commas plus one)

            '
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')'
            + --Count of the tables requested how many exist in the db

            '
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'','')) '
            + --only if the count matches (all the tables exist in the requested db)
then run the script

            '
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
                Insert [#ApInvoice]
                        ( [DatabaseName]
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_Reqs-AfterInvoices

```
                        , [Supplier]
                        , [Invoice]
                        , [InvoiceDate]
                        , [PostCurrency]
                        )
                SELECT [DatabaseName]=@DBCode
                     , [AI].[Supplier]
                     , [AI].[Invoice]
                     , [AI].[InvoiceDate]
                     , [AI].[PostCurrency] FROM [ApInvoice] As [AI]

                Insert [#ApSupplier]
                        ( [DatabaseName]
                        , [Supplier]
                        , [SupplierName]
                        )
                SELECT [DatabaseName]=@DBCode
                     , [AS].[Supplier]
                     , [AS].[SupplierName] FROM [ApSupplier] As [AS]
            End
    End';

--Enable this function to check script changes (try to run script directly against
db manually)

--Print @SQL


--execute script against each db, populating the base tables

        Exec [Process].[ExecForEachDB] @cmd = @SQLReqs;
        Exec [Process].[ExecForEachDB] @cmd = @SQLPorMasters;
        Exec [Process].[ExecForEachDB] @cmd = @SQLGrns;
        Exec [Process].[ExecForEachDB] @cmd = @SQLAps;

--define the results you want to return

        Create Table [#Results]
            (
              [DatabaseName] Varchar(150)        collate Latin1_General_BIN
            , [PurchaseOrder] Varchar(20)        collate Latin1_General_BIN
            , [Requisition] Varchar(10)            collate Latin1_General_BIN
            , [DateReqnRaised] Date
            , [OrderEntryDate] Date
            , [ApprovedDate] Date
            , [Grn] Varchar(20)                    collate Latin1_General_BIN
            , [Invoice] Varchar(20)                collate Latin1_General_BIN
            , [InvoiceDate] Date
            , [Supplier] Varchar(15)             collate Latin1_General_BIN
            , [SupplierName] Varchar(50)         collate Latin1_General_BIN
            , [StockCode] Varchar(30)             collate Latin1_General_BIN
            , [StockDescription] Varchar(50)     collate Latin1_General_BIN
            , [PostCurrency] Varchar(10)         collate Latin1_General_BIN
            , [MOrderQty] Numeric(20 , 8)
            , [MPrice] Numeric(20 , 2)
            , [MForeignPrice] Numeric(20 , 2)
            , [Warehouse] Varchar(10)             collate Latin1_General_BIN
```

Author: Johnson, Chris

```
            );

--Placeholder to create indexes as required


--script to combine base data and insert into results table

        Insert  [#Results]
                ( [DatabaseName]
                , [PurchaseOrder]
                , [Requisition]
                , [DateReqnRaised]
                , [OrderEntryDate]
                , [ApprovedDate]
                , [Grn]
                , [Invoice]
                , [InvoiceDate]
                , [Supplier]
                , [SupplierName]
                , [StockCode]
                , [StockDescription]
                , [PostCurrency]
                , [MOrderQty]
                , [MPrice]
                , [MForeignPrice]
                , [Warehouse]
                )
                Select  Distinct
                        [RH].[DatabaseName]
                      , [PMD].[PurchaseOrder]
                      , [RH].[Requisition]
                      , [RH].[DateReqnRaised]
                      , [PMH].[OrderEntryDate]
                      , [RD].[ApprovedDate]
                      , [GD].[Grn]
                      , [AI].[Invoice]
                      , [AI].[InvoiceDate]
                      , [AI].[Supplier]
                      , [AS].[SupplierName]
                      , [RD].[StockCode]
                      , [RD].[StockDescription]
                      , [AI].[PostCurrency]
                      , [PMD].[MOrderQty]
                      , [PMD].[MPrice]
                      , [PMD].[MForeignPrice]
                      , [GD].[Warehouse]
                From    [#ReqHeader] As [RH]
                        Left Join [#ReqDetail] As [RD] On [RD].[Requisition] =
[RH].[Requisition]
                                                          And [RD].[DatabaseName] =
[RH].[DatabaseName]
                        Left Join [#PorMasterDetail] As [PMD] On
[PMD].[MRequisition] = [RH].[Requisition]
                                                            And [PMD].[Line] =
[RD].[Line]
                                                            And [PMD].[Database-
Name] = [RD].[DatabaseName]
                        Left Join [#PorMasterHdr] As [PMH] On [PMH].[PurchaseOrder]
```

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_Reqs-AfterInvoices

```
= [PMD].[PurchaseOrder]
                                                                And [PMH].[Database-
Name] = [PMD].[DatabaseName]
                        Left Join [#GrnDetails] As [GD] On [GD].[PurchaseOrder] =
[PMH].[PurchaseOrder]
                                                            And [GD].[DatabaseName] =
[PMH].[DatabaseName]
                        Left Join [#GrnMatching] As [GM] On [GM].[Supplier] =
[GD].[Supplier]
                                                             And [GM].[Grn] =
[GD].[Grn]
                                                             And [GM].[DatabaseName]
= [GD].[DatabaseName]
                        Left Join [#ApInvoice] As [AI] On [AI].[Supplier] =
[GM].[Supplier]
                                                           And [AI].[Invoice] =
[GM].[Invoice]
                                                           And [AI].[DatabaseName] =
[GM].[DatabaseName]
                        Left Join [#ApSupplier] As [AS] On [AS].[Supplier] =
[AI].[Supplier]
                                                             And [AS].[DatabaseName] =
[AI].[DatabaseName]
                Where   [AI].[InvoiceDate] < [RH].[DateReqnRaised];

--return results

        Select  [R].[DatabaseName]
              , [R].[Requisition]
              , [R].[PurchaseOrder]
              , [R].[DateReqnRaised]
              , [R].[OrderEntryDate]
              , [R].[ApprovedDate]
              , [R].[Grn]
              , [R].[Invoice]
              , [R].[InvoiceDate]
              , [R].[Supplier]
              , [R].[SupplierName]
              , [R].[StockCode]
              , [R].[StockDescription]
              , [R].[PostCurrency]
              , [R].[MOrderQty]
              , [R].[MPrice]
              , [R].[MForeignPrice]
              , [R].[Warehouse]
              , [CN].[CompanyName]
        From    [#Results] As [R]
                Left Join [BlackBox].[Lookups].[CompanyNames] As [CN] On
[CN].[Company] = [R].[DatabaseName];


    End;

GO
EXEC sp_addextendedproperty N'MS_Description', N'list of requisitions raised post
invoices appearing in system', 'SCHEMA', N'Report', 'PROCEDURE', N'UspResults_Reqs-
AfterInvoices', NULL, NULL
GO
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_Reqs-AfterInvoices

Uses

[Lookups].[CompanyNames]

[Process].[ExecForEachDB]

[Process].[UspInsert_RedTagLogs]

[Report]

Author: Johnson, Chris

[Report].[UspResults_RequisitionStatus]

MS_Description

list of requisitions with status

Parameters

Name	Data Type	Max Length (Bytes)
@Company	varchar(max)	max
@RedTagType	char	1
@RedTagUse	varchar(500)	500

SQL Script

```
CREATE Proc [Report].[UspResults_RequisitionStatus]
    (
      @Company Varchar(Max)
    , @RedTagType Char(1)
    , @RedTagUse Varchar(500)
    )
As
    Begin
/*
Template designed by Chris Johnson, Prometic Group September 2015
Stored procedure set out to query multiple databases with the same information and
return it in a collated format
List of all requisitions and their statuses
--Exec [Report].[UspResults_RequisitionStatus]    @Company = '10' -- varchar(max)
*/
        Set NoCount Off;
        If IsNumeric(@Company) = 0
            Begin
                Select  @Company = Upper(@Company);
            End;

--remove nocount on to speed up query

        Set NoCount On;

    --Red tag

        Declare @RedTagDB Varchar(255)= Db_Name();
        Exec [Process].[UspInsert_RedTagLogs] @StoredProcDb = 'BlackBox' ,
            @StoredProcSchema = 'Report' ,
            @StoredProcName = 'UspResults_RequisitionStatus' ,
            @UsedByType = @RedTagType , @UsedByName = @RedTagUse ,
            @UsedByDb = @RedTagDB;


--list the tables that are to be pulled back from each DB - if they are not found
the script will not be run against that db
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-RequisitionStatus

```
        Declare @ListOfTables Varchar(Max) = 'ReqHeader,ReqDetail,ApSupplier';

--create temporary tables to be pulled from different databases, including a column
to id

        Create Table [#ReqHeader]
            (
              [DatabaseName] Varchar(150)        collate latin1_general_bin
            , [Requisition] Varchar(35)            collate latin1_general_bin
            );
        Create Table [#ReqDetail]
            (
              [DatabaseName] Varchar(150)        collate latin1_general_bin
            , [Buyer] Varchar(20)                collate latin1_general_bin
            , [CurrentHolder] Varchar(20)        collate latin1_general_bin
            , [DateReqnRaised] DateTime2
            , [DueDate] DateTime2
            , [Line] Int
            , [OrderQty] Numeric(20 , 6)
            , [Originator] Varchar(20)            collate latin1_general_bin
            , [Price] Numeric(18 , 3)
            , [StockCode] Varchar(35)            collate latin1_general_bin
            , [StockDescription] Varchar(150)    collate latin1_general_bin
            , [SupCatalogueNum] Varchar(50)        collate latin1_general_bin
            , [ReqnStatus] Varchar(10)           collate latin1_general_bin
            , [Requisition] Varchar(35)            collate latin1_general_bin
            , [Supplier] Varchar(35)            collate latin1_general_bin
            );
        Create Table [#ApSupplier]
            (
              [DatabaseName] Varchar(150)        collate latin1_general_bin
            , [Supplier] Varchar(35)            collate latin1_general_bin
            , [SupplierName] Varchar(150)        collate latin1_general_bin
            );

--create script to pull data from each db into the tables

        Declare @SQL Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end'
            + --Only query DBs beginning SysProCompany

            '
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN'
            + --only companies selected in main run, or if companies selected then
all

            '
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT'
            + --count number of tables requested (number of commas plus one)
```

Author: Johnson, Chris

```
            '
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')'
            + --Count of the tables requested how many exist in the db

            '
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'','')) '
            + --only if the count matches (all the tables exist in the requested db)
then run the script

            '
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
                Insert #ReqHeader
                    ( DatabaseName, Requisition )
            SELECT DatabaseName  = @DBCode
                 , Requisition FROM ReqHeader

            Insert #ReqDetail
                    ( DatabaseName, Buyer
                    , CurrentHolder, DateReqnRaised
                    , DueDate, Line
                    , OrderQty, Originator
                    , Price, StockCode
                    , StockDescription, SupCatalogueNum
                    , ReqnStatus, Requisition
                    , Supplier
                    )
            SELECT DatabaseName = @DBCode
                 , Buyer, CurrentHolder
                 , DateReqnRaised, DueDate
                 , Line, OrderQty
                 , Originator, Price
                 , StockCode, StockDescription
                 , SupCatalogueNum, ReqnStatus
                 , Requisition, Supplier FROM ReqDetail

            Insert #ApSupplier
                    ( DatabaseName, Supplier, SupplierName)
            SELECT DatabaseName = @DBCode, Supplier, SupplierName FROM ApSupplier
            End
    End';


                Exec [Process].[ExecForEachDB] @cmd = @SQL;
--execute script against each db, populating the base tables



--define the results you want to return

        Create Table [#ResultsReqStatus]
            (
              [DatabaseName] Varchar(150)        collate latin1_general_bin
            , [SupplierName] Varchar(150)        collate latin1_general_bin
```

Author: Johnson, Chris

```
            , [Buyer] Varchar(150)                collate latin1_general_bin
            , [CurrentHolder] Varchar(150)        collate latin1_general_bin
            , [DateReqnRaised] DateTime2
            , [DueDate] DateTime2
            , [Line] Varchar(10)                collate latin1_general_bin
            , [OrderQty] Float
            , [Originator] Varchar(150)            collate latin1_general_bin
            , [Price] Numeric(20 , 3)
            , [ReqStatus] Varchar(150)            collate latin1_general_bin
            , [StockCode] Varchar(35)            collate latin1_general_bin
            , [StockDescription] Varchar(150)    collate latin1_general_bin
            , [SupCatalogueNum] Varchar(50)        collate latin1_general_bin
            , [Requisition] Varchar(35)            collate latin1_general_bin
            , [CompanyName] Varchar(150)        collate latin1_general_bin
            );

--Placeholder to create indexes as required


--script to combine base data and insert into results table

        Insert  [#ResultsReqStatus]
                ( [DatabaseName]
                , [SupplierName]
                , [Buyer]
                , [CurrentHolder]
                , [DateReqnRaised]
                , [DueDate]
                , [Line]
                , [OrderQty]
                , [Originator]
                , [Price]
                , [ReqStatus]
                , [StockCode]
                , [StockDescription]
                , [SupCatalogueNum]
                , [Requisition]
                , [CompanyName]
                )
                Select  [CN].[Company]
                      , [APS].[SupplierName]
                      , [RD].[Buyer]
                      , [RD].[CurrentHolder]
                      , [RD].[DateReqnRaised]
                      , [RD].[DueDate]
                      , [RD].[Line]
                      , [RD].[OrderQty]
                      , [RD].[Originator]
                      , [RD].[Price]
                      , [ReqStatus] = [RS].[ReqnStatusDescription]
                      , [RD].[StockCode]
                      , [RD].[StockDescription]
                      , [RD].[SupCatalogueNum]
                      , [RH].[Requisition]
                      , [CN].[CompanyName]
                From    [BlackBox].[Lookups].[CompanyNames] As [CN]
                        Left Join [#ReqHeader] [RH]
```

Author: Johnson, Chris

```
                            On [CN].[Company] = [RH].[DatabaseName] Collate Latin1_-
General_BIN
                        Left Join [#ReqDetail] [RD]
                            On [RD].[Requisition] = [RH].[Requisition]
                               And [RD].[DatabaseName] = [RH].[DatabaseName]
                        Left Join [#ApSupplier] [APS]
                            On [APS].[Supplier] = [RD].[Supplier]
                               And [APS].[DatabaseName] = [RD].[DatabaseName]
                        Left Join [BlackBox].[Lookups].[ReqnStatus] [RS]
                            On [RS].[ReqnStatusCode] = [RD].[ReqnStatus] Collate
Latin1_General_BIN
                               And [RS].[Company] = [RD].[DatabaseName] Collate
Latin1_General_BIN
                Where   [CN].[Company] In (
                        Select  [USS].[Value]
                        From    [dbo].[udf_SplitString](@Company , ',') [USS] )
                        Or @Company = 'ALL';



--return results

        Select  [Company] = [RRS].[DatabaseName]
              , [RRS].[SupplierName]
              , [Buyer] = Case When [RRS].[Buyer] = '' Then 'Blank'
                               Else Coalesce([RRS].[Buyer] , 'Blank')
                          End
              , [RRS].[CurrentHolder]
              , [DateReqnRaised] = Cast([RRS].[DateReqnRaised] As Date)
              , [DueDate] = Cast([RRS].[DueDate] As Date)
              , [RRS].[Line]
              , [RRS].[OrderQty]
              , [RRS].[Originator]
              , [RRS].[Price]
              , [ReqStatus] = Coalesce([RRS].[ReqStatus] , 'No Status')
              , [RRS].[StockCode]
              , [RRS].[StockDescription]
              , [RRS].[SupCatalogueNum]
              , [RRS].[Requisition]
              , [RRS].[CompanyName]
        From    [#ResultsReqStatus] [RRS];

    End;

GO
EXEC sp_addextendedproperty N'MS_Description', N'list of requisitions with status',
'SCHEMA', N'Report', 'PROCEDURE', N'UspResults_RequisitionStatus', NULL, NULL
GO
```

Uses

[Lookups].[CompanyNames]
[Lookups].[ReqnStatus]
[Process].[ExecForEachDB]
[Process].[UspInsert_RedTagLogs]
[dbo].[udf_SplitString]

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-RequisitionStatus

[Report]

Author: Johnson, Chris

[Report].[UspResults_RequisitionUsers]

MS_Description

list of requisitions users

Parameters

Name	Data Type	Max Length (Bytes)
@Company	varchar(max)	max
@RedTagType	char	1
@RedTagUse	varchar(500)	500

SQL Script

```
CREATE Proc [Report].[UspResults_RequisitionUsers]
    (
      @Company Varchar(Max)
    , @RedTagType Char(1)
    , @RedTagUse Varchar(500)
    )
As
    Begin
/*
Template designed by Chris Johnson, Prometic Group September 2015
Stored procedure set out to query multiple databases with the same information and
return it in a collated format
List of all requisition users
*/
        If IsNumeric(@Company) = 0
            Begin
                Select  @Company = Upper(@Company);
            End;


--remove nocount on to speed up query

        Set NoCount On;
--Red tag

        Declare @RedTagDB Varchar(255)= Db_Name();
        Exec [Process].[UspInsert_RedTagLogs] @StoredProcDb = 'BlackBox' ,
            @StoredProcSchema = 'Report' ,
            @StoredProcName = 'UspResults_RequisitionUsers' ,
            @UsedByType = @RedTagType , @UsedByName = @RedTagUse ,
            @UsedByDb = @RedTagDB;


--list the tables that are to be pulled back from each DB - if they are not found
the script will not be run against that db

        Declare @ListOfTables Varchar(Max) = 'ReqUser,ReqGroup,ReqGroupAuthority';
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-RequisitionUsers

```
--create temporary tables to be pulled from different databases, including a column
to id

        Create Table [#ReqUser]
            (
              [DatabaseName] Varchar(150)         Collate Latin1_General_BIN
            , [UserCode] Varchar(20)             Collate Latin1_General_BIN
            , [UserName] Varchar(50)             Collate Latin1_General_BIN
            , [AuthorityLevel] Char(1)             Collate Latin1_General_BIN
            , [CanAddReqn] Char(1)                 Collate Latin1_General_BIN
            , [CanAddReqnDetails] Char(1)         Collate Latin1_General_BIN
            , [CanChgReqnDetails] Char(1)         Collate Latin1_General_BIN
            , [CanApproveReqn] Char(1)             Collate Latin1_General_BIN
            , [CanCreateOrder] Char(1)             Collate Latin1_General_BIN
            , [RequisitionGroup] Varchar(6)         Collate Latin1_General_BIN
            , [AdminStopFlag] Char(1)             Collate Latin1_General_BIN
            , [MaxApproveValue] Float
            );

        Create Table [#ReqGroup]
            (
              [DatabaseName] Varchar(150)         Collate Latin1_General_BIN
            , [RequisitionGroup] Varchar(6)         Collate Latin1_General_BIN
            , [GroupDescription] Varchar(50)     Collate Latin1_General_BIN
            );

        Create Table [#ReqGroupAuthority]
            (
              [DatabaseName] Varchar(150)         Collate Latin1_General_BIN
            , [RequisitionGroup] Varchar(6)         Collate Latin1_General_BIN
            , [MaxApproveValue] Float
            , [UserForPorder] Varchar(20)         Collate Latin1_General_BIN
            , [ProductClass] Varchar(20)         Collate Latin1_General_BIN
            );

--create script to pull data from each db into the tables

        Declare @SQL1 Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables
            + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'',''))
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-RequisitionUsers

```
                Insert #ReqUser
                        ( DatabaseName
                        , UserCode
                        , UserName
                        , AuthorityLevel
                        , CanAddReqn
                        , CanAddReqnDetails
                        , CanChgReqnDetails
                        , CanApproveReqn
                        , CanCreateOrder
                        , RequisitionGroup
                        , AdminStopFlag
                        , [MaxApproveValue]
                        )
                Select @DBCode
                    ,UserCode
                    ,UserName
                    ,AuthorityLevel
                    ,CanAddReqn
                    ,CanAddReqnDetails
                    ,CanChgReqnDetails
                    ,CanApproveReqn
                    ,CanCreateOrder
                    ,RequisitionGroup
                    ,AdminStopFlag
                    , [MaxApproveValue]
                 FROM [ReqUser]
            End
    End';
        Declare @SQL2 Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables
            + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'',''))
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
                Insert #ReqGroup
                    ( DatabaseName
                    , RequisitionGroup
                    , GroupDescription
                    )
                SELECT @DBCode
                    ,RequisitionGroup
                    , GroupDescription
                FROM [ReqGroup]
```

Author: Johnson, Chris

```
            End
    End';
        Declare @SQL3 Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables
            + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'',''))
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
                Insert #ReqGroupAuthority
                        ( DatabaseName
                        , RequisitionGroup
                        , MaxApproveValue
                        , UserForPorder
                        , [ProductClass]
                        )
                Select @DBCode
                     , RequisitionGroup
                     , MaxApproveValue
                     , UserForPorder
                     , [ProductClass]
                FROM  dbo.ReqGroupAuthority
            End
    End';

--Enable this function to check script changes (try to run script directly against
db manually)

--Print @SQL


--execute script against each db, populating the base tables

        Exec [Process].[ExecForEachDB] @cmd = @SQL1;
        Exec [Process].[ExecForEachDB] @cmd = @SQL2;
        Exec [Process].[ExecForEachDB] @cmd = @SQL3;

--define the results you want to return

        Create Table [#Results]
            (
              [DatabaseName] Varchar(150)         Collate Latin1_General_BIN
            , [UserCode] Varchar(20)             Collate Latin1_General_BIN
            , [UserName] Varchar(50)             Collate Latin1_General_BIN
            , [AuthorityLevel] Char(1)             Collate Latin1_General_BIN
```

```
            , [CanAddReqn] Char(1)                  Collate Latin1_General_BIN
            , [CanAddReqnDetails] Char(1)          Collate Latin1_General_BIN
            , [CanChgReqnDetails] Char(1)          Collate Latin1_General_BIN
            , [CanApproveReqn] Char(1)              Collate Latin1_General_BIN
            , [CanCreateOrder] Char(1)              Collate Latin1_General_BIN
            , [AdminStopFlag] Char(1)              Collate Latin1_General_BIN
            , [RequisitionGroup] Varchar(50)      Collate Latin1_General_BIN
            , [GroupMaxApproveValue] Float
            , [UserMaxApproveValue] Float
            , [ProductClass] Varchar(20)          Collate Latin1_General_BIN
            );


--Placeholder to create indexes as required


--script to combine base data and insert into results table

        Insert  [#Results]
                ( [DatabaseName]
                , [UserCode]
                , [UserName]
                , [AuthorityLevel]
                , [CanAddReqn]
                , [CanAddReqnDetails]
                , [CanChgReqnDetails]
                , [CanApproveReqn]
                , [CanCreateOrder]
                , [AdminStopFlag]
                , [RequisitionGroup]
                , [GroupMaxApproveValue]
                , [UserMaxApproveValue]
                , [ProductClass]
                )
                Select  [RU].[DatabaseName]
                      , [RU].[UserCode]
                      , [RU].[UserName]
                      , [RU].[AuthorityLevel]
                      , [RU].[CanAddReqn]
                      , [RU].[CanAddReqnDetails]
                      , [RU].[CanChgReqnDetails]
                      , [RU].[CanApproveReqn]
                      , [RU].[CanCreateOrder]
                      , [RU].[AdminStopFlag]
                      , [RequisitionGroup] = [RG].[GroupDescription]
                      , [RGA].[MaxApproveValue]
                      , [RU].[MaxApproveValue]
                      , [RGA].[ProductClass]
                From    [#ReqUser] [RU]
                        Left Join [#ReqGroup] [RG]
                            On [RG].[DatabaseName] = [RU].[DatabaseName]
                               And [RG].[RequisitionGroup] = [RU].[RequisitionGroup]
                        Left Join [#ReqGroupAuthority] [RGA]
                            On [RGA].[DatabaseName] = [RG].[DatabaseName]
                               And [RGA].[RequisitionGroup] = [RU].[Requisition-
Group];
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-RequisitionUsers

```
--return results

        Select  [CN].[Company]
              , [CN].[CompanyName]
              , [CN].[ShortName]
              , [CN].[Currency]
              , [R].[UserCode]
              , [R].[UserName]
              , [R].[AuthorityLevel]
              , [R].[CanAddReqn]
              , [R].[CanAddReqnDetails]
              , [R].[CanChgReqnDetails]
              , [R].[CanApproveReqn]
              , [R].[CanCreateOrder]
              , [ProductClass] = Case When Coalesce([R].[ProductClass] , '') = ''
                                      Then 'N/A'
                                      Else [R].[ProductClass]
                                 End
              , [RequisitionGroup] = Case When [R].[RequisitionGroup] = ''
                                          Then 'No Group'
                                          When [R].[RequisitionGroup] Is Null
                                          Then 'No Group'
                                          Else [R].[RequisitionGroup]
                                     End
              , [AdminStopFlag] = Case When [R].[AdminStopFlag] = '' Then Null
                                       Else [R].[AdminStopFlag]
                                  End
              , [GroupMaxApproveValue] = Case When [R].[GroupMaxApproveValue] = ''
                                              Then 0
                                              When [R].[GroupMaxApproveValue] Is
Null
                                              Then 0
                                              Else [R].[GroupMaxApproveValue]
                                         End
              , [UserMaxApproveValue] = Case When [R].[UserMaxApproveValue] = ''
                                             Then 0
                                             When [R].[UserMaxApproveValue] Is Null
                                             Then 0
                                             Else [R].[UserMaxApproveValue]
                                        End
        From    [#Results] As [R]
                Left Join [BlackBox].[Lookups].[CompanyNames] [CN]
                    On [R].[DatabaseName] = [CN].[Company];

    End;

GO
EXEC sp_addextendedproperty N'MS_Description', N'list of requisitions users',
'SCHEMA', N'Report', 'PROCEDURE', N'UspResults_RequisitionUsers', NULL, NULL
GO
```

Uses

[Lookups].[CompanyNames]
[Process].[ExecForEachDB]
[Process].[UspInsert_RedTagLogs]

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-RequisitionUsers

[Report]

Author: Johnson, Chris

[Report].[UspResults_ReservedLots]

MS_Description

list of reserved lots used (allocated blood)

Parameters

Name	Data Type	Max Length (Bytes)
@Company	varchar(10)	10
@StockCode	varchar(20)	20
@Job	varchar(50)	50
@RedTagType	char	1
@RedTagUse	varchar(500)	500

SQL Script

```
CREATE Proc [Report].[UspResults_ReservedLots]
    (
      @Company Varchar(10)
    , @StockCode Varchar(20)
    , @Job Varchar(50)
    , @RedTagType Char(1)
    , @RedTagUse Varchar(500)
    )
As /*
Template designed by Chris Johnson, Prometic Group September 2015
Stored procedure set out to query multiple databases with the same information and
return it in a collated format
--Exec Report.UspResults_ReservedLots  @Company    ='F',@StockCode
='000000000000005',@Job        ='000000000000012'
*/
    Set NoCount On;
    If IsNumeric(@Company) = 0
        Begin
            Select  @Company = Upper(@Company);
        End;

--Red tag

        Declare @RedTagDB Varchar(255)= Db_Name();
        Exec [Process].[UspInsert_RedTagLogs] @StoredProcDb = 'BlackBox' ,
            @StoredProcSchema = 'Report' ,
            @StoredProcName = 'UspResults_ReservedLots' ,
            @UsedByType = @RedTagType , @UsedByName = @RedTagUse ,
            @UsedByDb = @RedTagDB;

--Convert Job to varchar for querying DB

    Declare @JobVarchar Varchar(20);
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-ReservedLots

```
--Cater for number jobs

    Select  @JobVarchar = Case When IsNumeric(@Job) = 1
                               Then Right('000000000000000' + @Job , 15)
                               Else @Job
                          End;

--list the tables that are to be pulled back from each DB - if they are not found
the script will not be run against that db

    Declare @ListOfTables Varchar(Max) = 'WipMasterSub,TblApTerms';

--Create table to capture results

    Create Table [#WipAllMatLot]
        (
          [DatabaseName] Varchar(150)    Collate Latin1_General_BIN
        , [Job] Varchar(20)                Collate Latin1_General_BIN
        , [StockCode] Varchar(30)        Collate Latin1_General_BIN
        , [Lot] Varchar(50)                Collate Latin1_General_BIN
        , [Bin] Varchar(20)                Collate Latin1_General_BIN
        , [Warehouse] Varchar(20)        Collate Latin1_General_BIN
        , [QtyReserved] Numeric(20 , 8)
        , [QtyIssued] Numeric(20 , 8)
        );
    Create Table [#InvMaster]
        (
          [DatabaseName] Varchar(150)    Collate Latin1_General_BIN
        , [Description] Varchar(50)        Collate Latin1_General_BIN
        , [StockCode] Varchar(20)        Collate Latin1_General_BIN
        , [PartCategory] Char(1)        Collate Latin1_General_BIN
        , [IssMultLotsFlag] Char(1)        Collate Latin1_General_BIN
        , [StockUom] Varchar(10)        Collate Latin1_General_BIN
        );
    Create Table [#CusLot]
        (
          [DatabaseName] Varchar(150)        Collate Latin1_General_BIN
        , [Lot] Varchar(50)                    Collate Latin1_General_BIN
        , [StockCode] Varchar(30)            Collate Latin1_General_BIN
        , [BleedNumber] Varchar(20)            Collate Latin1_General_BIN
        , [DonorNumber] Varchar(20)            Collate Latin1_General_BIN
        , [VendorBatchNumber] Varchar(50)    Collate Latin1_General_BIN
        , [OldLotNumber] Varchar(20)        Collate Latin1_General_BIN
        , [BleedDate] Varchar(15)            Collate Latin1_General_BIN
        );

--create script to pull data from each db into the tables

    Declare @SQLInvMaster Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end'
        + --Only query DBs beginning SysProCompany

        '
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-ReservedLots

```
    BEGIN'
        + --only companies selected in main run, or if companies selected then all

        '
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
        + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT'
        + --count number of tables requested (number of commas plus one)

        '
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')'
        + --Count of the tables requested how many exist in the db

        '
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'','')) '
        + --only if the count matches (all the tables exist in the requested db)
then run the script

        '
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
            Insert [#InvMaster]
                    ( [DatabaseName]
                    , [StockCode]
                    , [PartCategory]
                    , [IssMultLotsFlag]
                    , [StockUom]
                    , Description
                    )
            SELECT [DatabaseName]=@DBCode
                 , [StockCode]
                 , [PartCategory]
                 , [IssMultLotsFlag]
                 , [StockUom]
                 , Description
            FROM [InvMaster] As [im]
            where [StockCode] = ''' + @StockCode + '''
            End
    End';
    Declare @SQLWipAllMatLot Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end'
        + --Only query DBs beginning SysProCompany

        '
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN'
        + --only companies selected in main run, or if companies selected then all

        '
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-ReservedLots

```
        + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT'
        + --count number of tables requested (number of commas plus one)

        '
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')'
        + --Count of the tables requested how many exist in the db

        '
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'','')) '
        + --only if the count matches (all the tables exist in the requested db)
then run the script

        '
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
            Insert [#WipAllMatLot]
                    ( [DatabaseName]
                    , [Job]
                    , [StockCode]
                    , [Lot]
                    , [Bin]
                    , [QtyReserved]
                    , [QtyIssued]
                    , Warehouse
                    )
            SELECT @DBCode
                    , [Job]
                    , [StockCode]
                    , [Lot]
                    , [Bin]
                    , [QtyReserved]
                    , [QtyIssued]
                    , Warehouse
            From [WipAllMatLot] As [waml]
            where Job = ''' + @Job + '''
            and StockCode = ''' + @StockCode + '''
            End
    End';
    Declare @SQLCusLot Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end'
        + --Only query DBs beginning SysProCompany

        '
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN'
        + --only companies selected in main run, or if companies selected then all

        '
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-ReservedLots

```
        + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT'
        + --count number of tables requested (number of commas plus one)

        '
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')'
        + --Count of the tables requested how many exist in the db

        '
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'','')) '
        + --only if the count matches (all the tables exist in the requested db)
then run the script

        '
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
            Declare @SQLSub Varchar(Max) =''Insert  #CusLot
                        ( DatabaseName
                        , Lot
                        , StockCode
                        , BleedNumber
                        , DonorNumber
                        , VendorBatchNumber
                        , OldLotNumber
                        , BleedDate
                        )
                        Select  ''''''+@DBCode+''''''
                                , Lot
                              , StockCode
                              , BleedNumber
                              , DonorNumber
                              , VendorBatchNumber
                              , OldLotNumber
                              , BleedDate
                        From    [dbo].[CusLot+]
                        where StockCode=''''' + @StockCode + '''''''
    Exec (@SQLSub)
            End
    End';
    --Print 1

    Exec [Process].[ExecForEachDB] @cmd = @SQLInvMaster;
    --Print 7

    Exec [Process].[ExecForEachDB] @cmd = @SQLWipAllMatLot;
    --Print 9

    Exec [Process].[ExecForEachDB] @cmd = @SQLCusLot;
    Print @SQLCusLot;


    Select  [waml].[Job]
          , [waml].[StockCode]
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-ReservedLots

```
          , [IM].[Description]
          , [ReservedLot] = [waml].[Lot]
          , [ReservedLotBin] = [waml].[Bin]
          , [ReservedLotWarehouse] = [waml].[Warehouse]
          , [ReservedLotQtyReserved] = [waml].[QtyReserved]
          , [ReservedLotQtyIssued] = [waml].[QtyIssued]
          , [ReservedLotBleedNumber] = [CL].[BleedNumber]
          , [ReservedLotDonorNumber] = [CL].[DonorNumber]
          , [ReservedLotVendorBatchNumber] = [CL].[VendorBatchNumber]
          , [ReservedLotOldLotNumber] = [CL].[OldLotNumber]
          , [ReservedLotBleedDate] = [CL].[BleedDate]
    From    [#WipAllMatLot] As [waml]
            Left Join [#CusLot] As [CL] On [CL].[DatabaseName] = [waml].[Database-
Name]
                                       And [CL].[Lot] = [waml].[Lot]
                                       And [CL].[StockCode] = [waml].[StockCode]
            Left Join [#InvMaster] As [IM] On [IM].[DatabaseName] = [waml].[Database-
Name]
                                          And [IM].[StockCode] = [waml].[StockCode]
    Where   [waml].[Job] = @Job
            And [waml].[StockCode] = @StockCode;


--tidy up


    Drop Table [#WipAllMatLot];
    Drop Table [#InvMaster];
    Drop Table [#CusLot];




GO
EXEC sp_addextendedproperty N'MS_Description', N'list of reserved lots used
(allocated blood)', 'SCHEMA', N'Report', 'PROCEDURE', N'UspResults_ReservedLots',
NULL, NULL
GO
```

Uses

[Process].[ExecForEachDB]
[Process].[UspInsert_RedTagLogs]
[Report]

Author: Johnson, Chris

[Report].[UspResults_SalesOrderKPI]

MS_Description

data for sales order KPI report

Parameters

Name	Data Type	Max Length (Bytes)
@Company	varchar(max)	max
@RedTagType	char	1
@RedTagUse	varchar(500)	500

SQL Script

```
CREATE Proc [Report].[UspResults_SalesOrderKPI]
    (
      @Company Varchar(Max)
    , @RedTagType Char(1)
    , @RedTagUse Varchar(500)
    )
As
    Begin
        If IsNumeric(@Company) = 0
            Begin
                Select  @Company = Upper(@Company);
            End;

Set NoCount on
--Red tag

        Declare @RedTagDB Varchar(255)= Db_Name();
        Exec [Process].[UspInsert_RedTagLogs] @StoredProcDb = 'BlackBox' ,
            @StoredProcSchema = 'Report' ,
            @StoredProcName = 'UspResults_SalesOrderKPI' ,
            @UsedByType = @RedTagType , @UsedByName = @RedTagUse ,
            @UsedByDb = @RedTagDB;

--list the tables that are to be pulled back from each DB - if they are not found
the script will not be run against that db

        Declare @ListOfTables Varchar(Max) = 'MdnMasterRep,CusSorMaster+';

--create temporary tables to be pulled from different databases, including a column
to id

        Create Table [#MdnMasterRep]
            (
              [DatabaseName] Varchar(150)            collate latin1_general_bin
            , [PlannedDeliverDate] DateTime
            , [ActualDeliveryDate] DateTime
            , [SalesOrder] Varchar(20)                collate latin1_general_bin
```

Author: Johnson, Chris

```
            , [Invoice] Varchar(20)                    collate latin1_general_bin
            , [DispatchComments1] Varchar(100)         collate latin1_general_bin
            , [DispatchComments2] Varchar(100)         collate latin1_general_bin
            , [DispatchComments3] Varchar(100)         collate latin1_general_bin
            , [DispatchComments4] Varchar(100)         collate latin1_general_bin
            );
        Create Table [#SorMaster]
            (
              [DatabaseName] Varchar(150)            collate latin1_general_bin
            , [SalesOrder] Varchar(20)               collate latin1_general_bin
            , [OrderDate] DateTime
            , [EntrySystemDate] DateTime
            , [ReqShipDate] DateTime
            , [CustomerPoNumber] Varchar(30)         collate latin1_general_bin
            , [CustomerName] Varchar(50)             collate latin1_general_bin
            , [Customer] Varchar(15)                 collate latin1_general_bin
            , [NonMerchFlag] Char(1)                 collate latin1_general_bin
            , [OrderStatus] Char(1)                     collate latin1_general_bin
            , [Warehouse] Varchar(10)                collate latin1_general_bin
            );
        Create Table [#CusSorMasterPlus]
            (
              [DatabaseName] Varchar(150)            collate latin1_general_bin
            , [SalesOrder] Varchar(20)               collate latin1_general_bin
            , [AcceptedDate] DateTime
            );
        Create Table [#SorDetail]
            (
              [DatabaseName] Varchar(150)            collate latin1_general_bin
            , [SalesOrder] Varchar(20)               collate latin1_general_bin
            , [SalesOrderLine] Int
            , [MStockCode] Varchar(30)               collate latin1_general_bin
            , [MStockDes] Varchar(50)                collate latin1_general_bin
            , [MOrderQty] Numeric(20 , 8)
            , [LineType] Char(1)                     collate latin1_general_bin
            );
        Create Table [#MdnDetail]
            (
              [DatabaseName] Varchar(150)            collate latin1_general_bin
            , [DispatchNote] Varchar(20)             collate latin1_general_bin
            , [DispatchStatus] Char(1)               collate latin1_general_bin
            , [SalesOrder] Varchar(20)               collate latin1_general_bin
            , [SalesOrderLine] Int
            , [LineType] Char(1)                     collate latin1_general_bin
            , [JnlYear] Int
            );
        Create Table [#LotTransactions]
            (
              [DatabaseName] Varchar(150)            collate latin1_general_bin
            , [Lot] Varchar(50)                         collate latin1_general_bin
            , [Job] Varchar(20)                         collate latin1_general_bin
            , [SalesOrder] Varchar(20)               collate latin1_general_bin
            , [SalesOrderLine] Int
            , [StockCode] Varchar(30)                collate latin1_general_bin
            , [DispatchNote] Varchar(20)             collate latin1_general_bin
            , [Invoice] Varchar(20)                     collate latin1_general_bin
            , [NewWarehouse] Varchar(10)             collate latin1_general_bin
```

Author: Johnson, Chris

```
            , [JnlYear] Int
            , [TrnType] Char(1)                          collate latin1_general_bin
            , [Reference] Varchar(30)                   collate latin1_general_bin
            , [JobPurchOrder] Varchar(30)               collate latin1_general_bin
            );

--create script to pull data from each db into the tables

        Declare @SQLMdnMasterRep Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables
            + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'',''))
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
                Insert [#MdnMasterRep]
                        ( [DatabaseName]
                        , [PlannedDeliverDate]
                        , [ActualDeliveryDate]
                        , [SalesOrder]
                        , [Invoice]
                        , [DispatchComments1]
                        , [DispatchComments2]
                        , [DispatchComments3]
                        , [DispatchComments4]
                        )
                SELECT @DBCode
                     , [MMR].[PlannedDeliverDate]
                     , [MMR].[ActualDeliveryDate]
                     , [MMR].[SalesOrder]
                     , [MMR].[Invoice]
                     , [MMR].[DispatchComments1]
                     , [MMR].[DispatchComments2]
                     , [MMR].[DispatchComments3]
                     , [MMR].[DispatchComments4]
                FROM [MdnMasterRep] As [MMR]
            End
    End';
        Declare @SQLSorMaster Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-SalesOrderKPI

```
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables
            + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'',''))
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
            Insert  [#SorMaster]
                    ( [DatabaseName]
                    , [SalesOrder]
                    , [OrderDate]
                    , [EntrySystemDate]
                    , [ReqShipDate]
                    , [CustomerPoNumber]
                    , [CustomerName]
                    , [Customer]
                    , [NonMerchFlag]
                    , [OrderStatus]
                    , [Warehouse]
                    )
            Select  @DBCode
                  , [SM].[SalesOrder]
                  , [SM].[OrderDate]
                  , [SM].[EntrySystemDate]
                  , [SM].[ReqShipDate]
                  , [SM].[CustomerPoNumber]
                  , [SM].[CustomerName]
                  , [SM].[Customer]
                  , [SM].[NonMerchFlag]
                  , [SM].[OrderStatus]
                  , [SM].[Warehouse]
            From    [SorMaster] As [SM];
            End
    End';
        Declare @SQLCusSorMasterPlus Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables
            + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'',''))
```

Author: Johnson, Chris

```
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
             Declare @SQLSub Varchar(2000)= ''Insert  [#CusSorMasterPlus]
                        ( [DatabaseName]
                        , [SalesOrder]
                        , [AcceptedDate]
                        )
                        Select  ''''''+@DBCode+''''''
                  , [CSMP].[SalesOrder]
                  , [CSMP].[AcceptedDate]
            From    dbo.[CusSorMaster+] As [CSMP];'';


                Exec (@SQLSub);
            End
    End';
        Declare @SQLSorDetail Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables
            + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'',''))
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
            Insert [#SorDetail]
            ( [DatabaseName]
            , [SalesOrder]
            , [SalesOrderLine]
            , [MStockCode]
            , [MStockDes]
            , [MOrderQty]
            , [LineType]
            )
            SELECT @DBCode
                  , [SD].[SalesOrder]
                  , [SD].[SalesOrderLine]
                  , [SD].[MStockCode]
                  , [SD].[MStockDes]
                  , [SD].[MOrderQty]
                  , [SD].[LineType]
            FROM [SorDetail] As [SD]
            End
    End';
        Declare @SQLMdnDetail Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-SalesOrderKPI

```
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables
            + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'',''))
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
            Insert  [#MdnDetail]
            ( [DatabaseName]
            , [DispatchNote]
            , [DispatchStatus]
            , [SalesOrder]
            , [SalesOrderLine]
            , [LineType]
            , [JnlYear]
            )
            Select  @DBCode
                  , [MD].[DispatchNote]
                  , [MD].[DispatchStatus]
                  , [MD].[SalesOrder]
                  , [MD].[SalesOrderLine]
                  , [MD].[LineType]
                  , [MD].[JnlYear]
            From    [MdnDetail] As [MD];
            End
    End';
        Declare @SQLLotTransactions Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables
            + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'',''))
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
            Insert  [#LotTransactions]
            ( [DatabaseName]
```

Author: Johnson, Chris

```
            , [Lot]
            , [Job]
            , [SalesOrder]
            , [SalesOrderLine]
            , [StockCode]
            , [DispatchNote]
            , [Invoice]
            , [NewWarehouse]
            , [JnlYear]
            , [TrnType]
            , [Reference]
            , [JobPurchOrder]
            )
            Select  @DBCode
                  , [Lot] = case when [LT].[Lot]='''' then null
                                 else [LT].[Lot] end
                  , [LT].[Job]
                  , [LT].[SalesOrder]
                  , [LT].[SalesOrderLine]
                  , [LT].[StockCode]
                  , [LT].[DispatchNote]
                  , [LT].[Invoice]
                  , [LT].[NewWarehouse]
                  , [LT].[JnlYear]
                  , [LT].[TrnType]
                  , [Reference] = case when LT.[Reference]='''' then null
                                       else LT.[Reference] end
                  , [JobPurchOrder] = case when LT.[JobPurchOrder] ='''' then null
                                           else LT.[JobPurchOrder] end
            From    [LotTransactions] As [LT];
            End
    End';


--Enable this function to check script changes (try to run script directly against
db manually)

--Print @SQLCusSorMasterPlus


--execute script against each db, populating the base tables

        Exec [Process].[ExecForEachDB] @cmd = @SQLMdnMasterRep;
        Exec [Process].[ExecForEachDB] @cmd = @SQLSorMaster;
        Exec [Process].[ExecForEachDB] @cmd = @SQLCusSorMasterPlus;
        Exec [Process].[ExecForEachDB] @cmd = @SQLSorDetail;
        Exec [Process].[ExecForEachDB] @cmd = @SQLMdnDetail;
        Exec [Process].[ExecForEachDB] @cmd = @SQLLotTransactions;

--define the results you want to return

        Create Table [#Results]
            (
              [CompanyName] Varchar(300)            collate latin1_general_bin
            , [SalesOrder] Varchar(20)                collate latin1_general_bin
            , [SalesOrderLine] Int
            , [OrderDate] Date
```

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-SalesOrderKPI

```
            , [EntrySystemDate] Date
            , [ReqShipDate] Date
            , [AcceptedDate] Date
            , [CustomerPoNumber] Varchar(30)        collate latin1_general_bin
            , [CustomerName] Varchar(50)            collate latin1_general_bin
            , [Customer] Varchar(15)                collate latin1_general_bin
            , [Lot] Varchar(50)                        collate latin1_general_bin
            , [StockCode] Varchar(30)                collate latin1_general_bin
            , [StockDescription] Varchar(50)        collate latin1_general_bin
            , [OrderQty] Numeric(20 , 8)
            , [DispatchNote] Varchar(20)            collate latin1_general_bin
            , [DispatchStatus] Char(1)                collate latin1_general_bin
            , [PlannedDeliverDate] Date
            , [ActualDeliveryDate] Date
            , [DaysDiff] Int
            , [NonMerchFlag] Char(1)                collate latin1_general_bin
            , [OrderStatus] Char(1)                    collate latin1_general_bin
            , [Job] Varchar(20)                        collate latin1_general_bin
            , [DispatchComments] Varchar(500)        collate latin1_general_bin
            );

--Placeholder to create indexes as required


--script to combine base data and insert into results table

        Insert  [#Results]
                ( [CompanyName]
                , [SalesOrder]
                , [SalesOrderLine]
                , [OrderDate]
                , [EntrySystemDate]
                , [ReqShipDate]
                , [AcceptedDate]
                , [CustomerPoNumber]
                , [CustomerName]
                , [Customer]
                , [Lot]
                , [StockCode]
                , [StockDescription]
                , [OrderQty]
                , [DispatchNote]
                , [DispatchStatus]
                , [PlannedDeliverDate]
                , [ActualDeliveryDate]
                , [DaysDiff]
                , [NonMerchFlag]
                , [OrderStatus]
                , [Job]
                , [DispatchComments]
                )
                Select  [CN].[CompanyName]
                      , [SM].[SalesOrder]
                      , [SD].[SalesOrderLine]
                      , [OrderDate] = Cast([SM].[OrderDate] As Date)
                      , [EntrySystemDate] = Cast([SM].[EntrySystemDate] As Date)
                      , [ReqShipDate] = Cast([SM].[ReqShipDate] As Date)
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-SalesOrderKPI

```
                      , [AcceptedDate] = Cast([CSM].[AcceptedDate] As Date)
                      , [SM].[CustomerPoNumber]
                      , [SM].[CustomerName]
                      , [SM].[Customer]
                      , [LT].[Lot]
                      , [SD].[MStockCode]
                      , [SD].[MStockDes]
                      , [SD].[MOrderQty]
                      , [MD].[DispatchNote]
                      , [MD].[DispatchStatus]
                      , [PlannedDeliverDate] = Cast([MMR].[PlannedDeliverDate] As
Date)
                      , [ActualDeliveryDate] = Cast([MMR].[ActualDeliveryDate] As
Date)
                      , [DaysDiff] = DateDiff(Day , [MMR].[PlannedDeliverDate] ,
                                              [MMR].[ActualDeliveryDate])
                      , [SM].[NonMerchFlag]
                      , [SM].[OrderStatus]
                      , [LT3].[Job]
                      , [DispatchComments] = Coalesce([MMR].[DispatchComments1] ,
                                                      '') + Coalesce(Char(13)
                                                                     + Case
                                                                     When [MMR].[Dispatch-
Comments2] = ''
                                                                     Then Null
                                                                     Else [MMR].[Dispatch-
Comments2]
                                                                     End , '')
                        + Coalesce(Char(13)
                                   + Case When [MMR].[DispatchComments3] = ''
                                          Then Null
                                          Else [MMR].[DispatchComments3]
                                     End , '') + Coalesce(Char(13)
                                                          + Case
                                                              When [MMR].[Dispatch-
Comments4] = ''
                                                              Then Null
                                                              Else [MMR].[Dispatch-
Comments4]
                                                            End , '')
                from    [#SorMaster] As [SM]
                        Inner Join [#MdnMasterRep] As [MMR]
                            On [MMR].[SalesOrder] = [SM].[SalesOrder]
                               And [MMR].[DatabaseName] = [SM].[DatabaseName]
                        Left Join [#CusSorMasterPlus] As [CSM]
                            On [CSM].[SalesOrder] = [MMR].[SalesOrder]
                               And [CSM].[DatabaseName] = [MMR].[DatabaseName]
                        Inner Join [#SorDetail] As [SD]
                            On [SD].[SalesOrder] = [SM].[SalesOrder]
                               And [SD].[LineType] <> 5
                               And [SD].[DatabaseName] = [SM].[DatabaseName]
                        Inner Join [#MdnDetail] As [MD]
                            On [MD].[SalesOrder] = [SD].[SalesOrder]
                               And [MD].[SalesOrderLine] = [SD].[SalesOrderLine]
                               And [MD].[LineType] = [SD].[LineType]
                               And [MD].[DatabaseName] = [SD].[DatabaseName]
                        Left Join [#LotTransactions] As [LT]
                            On [LT].[SalesOrder] = [SD].[SalesOrder]
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-SalesOrderKPI

```
                                And [LT].[SalesOrderLine] = [SD].[SalesOrderLine]
                                And [LT].[StockCode] = [SD].[MStockCode]
                                And [LT].[DispatchNote] = [MD].[DispatchNote]
                                And [LT].[Invoice] = [MMR].[Invoice]
                                And [LT].[NewWarehouse] = [SM].[Warehouse]
                                And [LT].[JnlYear] = [MD].[JnlYear]
                                And [LT].[DatabaseName] = [MD].[DatabaseName]
                        Left Join ( Select Distinct
                                            [LT2].[Lot]
                                          , [LT2].[StockCode]
                                          , [Job] = Coalesce([LT2].[JobPurchOrder] ,
                                                             [LT2].[Job] ,
                                                             [LT2].[Reference])
                                          , [LT2].[DatabaseName]
                                    From    [#LotTransactions] As [LT2]
                                    Where   [LT2].[TrnType] = 'R'
                                      And Coalesce([LT2].[JobPurchOrder] ,
                                                   [LT2].[Job] ,
                                                   [LT2].[Reference] ,
                                                   '') <> ''
                                  ) [LT3]
                            On [LT].[Lot] = [LT3].[Lot]
                               And [LT].[StockCode] = [LT3].[StockCode]
                               And [LT3].[DatabaseName] = [LT].[DatabaseName]
                        Left Join [Lookups].[CompanyNames] As [CN]
                            On [SM].[DatabaseName] = [CN].[Company]
                Where   [SM].[OrderStatus] In ( '0' , '1' , '2' , '3' , '4' ,
                                                '8' , '9' )
                        And [LT].[Lot] Is Not Null
                Order By [SM].[SalesOrder] Asc;

Set NoCount Off
--return results

        Select  [CompanyName]
              , [SalesOrder]
              , [SalesOrderLine]
              , [OrderDate]
              , [EntrySystemDate]
              , [ReqShipDate]
              , [AcceptedDate]
              , [CustomerPoNumber]
              , [CustomerName]
              , [Customer]
              , [Lot]
              , [StockCode]
              , [StockDescription]
              , [OrderQty]
              , [DispatchNote]
              , [DispatchStatus]
              , [PlannedDeliverDate]
              , [ActualDeliveryDate]
              , [DaysDiff]
              , [NonMerchFlag]
              , [OrderStatus]
              , [Job]
              , [DispatchComments]
```

Author: Johnson, Chris

```
              , [WorkingDaysBetweenAcceptedAndDeliveryDate] = Case
                                                              When [AcceptedDate] Is
Null
                                                              Then Null
                                                              When [AcceptedDate] =
[ActualDeliveryDate]
                                                              Then 0
                                                              Else [Process].[Udf_-
WorkingDays]([AcceptedDate] ,
                                                              [ActualDeliveryDate] ,
                                                              'UK') - 1
                                                              End
        From    [#Results];

    End;

GO
EXEC sp_addextendedproperty N'MS_Description', N'data for sales order KPI report',
'SCHEMA', N'Report', 'PROCEDURE', N'UspResults_SalesOrderKPI', NULL, NULL
GO
```

Uses

[Lookups].[CompanyNames]
[Process].[ExecForEachDB]
[Process].[UspInsert_RedTagLogs]
[Process].[Udf_WorkingDays]
[Report]

Author: Johnson, Chris

[Report].[UspResults_SalesOrders]

MS_Description

list of sales orders

Parameters

Name	Data Type	Max Length (Bytes)
@Company	varchar(max)	max
@RedTagType	char	1
@RedTagUse	varchar(500)	500

SQL Script

```
CREATE Proc [Report].[UspResults_SalesOrders]
    (
      @Company Varchar(Max)
    , @RedTagType Char(1)
    , @RedTagUse Varchar(500)
    )
As
    Begin
/*
Template designed by Chris Johnson, Prometic Group September 2015
Stored procedure set out to query multiple databases with the same information and
return it in a collated format
List of all Sales orders with details
--exec [Report].[UspResults_SalesOrders]  10
*/
        If IsNumeric(@Company) = 0
            Begin
                Select  @Company = Upper(@Company);
            End;

--remove nocount on to speed up query

        Set NoCount On;
--Red tag

        Declare @RedTagDB Varchar(255)= Db_Name();
        Exec [Process].[UspInsert_RedTagLogs] @StoredProcDb = 'BlackBox' ,
            @StoredProcSchema = 'Report' ,
            @StoredProcName = 'UspResults_SalesOrders' ,
            @UsedByType = @RedTagType , @UsedByName = @RedTagUse ,
            @UsedByDb = @RedTagDB;
--list the tables that are to be pulled back from each DB - if they are not found
the script will not be run against that db

        Declare @ListOfTables Varchar(Max) = 'SorMaster,SorDetail,SalSalesperson';

--create temporary tables to be pulled from different databases, including a column
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-SalesOrders

```
to id

        Create Table [#SorMaster]
            (
              [DatabaseName] Varchar(150)        Collate Latin1_General_BIN
            , [SalesOrder] Varchar(35)            Collate Latin1_General_BIN
            , [CancelledFlag] Char(1)            collate latin1_general_bin
            , [Customer] Varchar(50)            Collate Latin1_General_BIN
            , [CustomerName] Varchar(255)        Collate Latin1_General_BIN
            , [OrderStatus] Varchar(5)            Collate Latin1_General_BIN
            , [Salesperson] Varchar(20)            Collate Latin1_General_BIN
            , [Branch] Varchar(10)                Collate Latin1_General_BIN
            , [CustomerPoNumber] Varchar(30)    Collate Latin1_General_BIN
            , [OrderDate] DateTime2
            , [EntrySystemDate] DateTime2
            , [ReqShipDate] DateTime2
            , [Currency] Varchar(5)                Collate Latin1_General_BIN
            );
        Create Table [#SorDetail]
            (
              [DatabaseName] Varchar(150)        Collate Latin1_General_BIN
            , [SalesOrder] Varchar(35)            Collate Latin1_General_BIN
            , [SalesOrderLine] Int
            , [LineType] Varchar(10)            Collate Latin1_General_BIN
            , [MStockCode] Varchar(35)            Collate Latin1_General_BIN
            , [MStockDes] Varchar(255)            Collate Latin1_General_BIN
            , [MOrderQty] Numeric(20 , 7)
            , [MOrderUom] Varchar(5)            Collate Latin1_General_BIN
            , [MPrice] Numeric(20 , 2)
            , [NComment] Varchar(100)            Collate Latin1_General_BIN
            , [NMscProductCls] Varchar(20)        Collate Latin1_General_BIN
            , [NMscChargeValue] Numeric(20 , 3)
            );
        Create Table [#SalSalesperson]
            (
              [DatabaseName] Varchar(150)        Collate Latin1_General_BIN
            , [Name] Varchar(50)                Collate Latin1_General_BIN
            , [Branch] Varchar(10)                Collate Latin1_General_BIN
            , [Salesperson] Varchar(20)            Collate Latin1_General_BIN
            );

--create script to pull data from each db into the tables

        Declare @SQL1 Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN'
            + --only companies selected in main run, or if companies selected then
all

            '
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables + '''
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-SalesOrders

```
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT'
            + --count number of tables requested (number of commas plus one)

            '
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')'
            + --Count of the tables requested how many exist in the db

            '
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'','')) '
            + --only if the count matches (all the tables exist in the requested db)
then run the script

            '
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
                Insert  [#SorMaster]
                ( [DatabaseName]
                , [SalesOrder]
                , [CancelledFlag]
                , [Customer]
                , [CustomerName]
                , [OrderStatus]
                , [Salesperson]
                , [Branch]
                , [CustomerPoNumber]
                , [OrderDate]
                , [EntrySystemDate]
                , [ReqShipDate]
                , [Currency]
                )
                Select
                    [DatabaseName] = @DBCode
                  , [sm].[SalesOrder]
                  , [sm].[CancelledFlag]
                  , [sm].[Customer]
                  , [sm].[CustomerName]
                  , [sm].[OrderStatus]
                  , [sm].[Salesperson]
                  , [sm].[Branch]
                  , [sm].[CustomerPoNumber]
                  , [sm].[OrderDate]
                  , [sm].[EntrySystemDate]
                  , [sm].[ReqShipDate]
                  , [sm].[Currency]
                From
                    [SorMaster] As [sm];
            End
    End';
        Declare @SQL2 Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-SalesOrders

```
    BEGIN'
            + --only companies selected in main run, or if companies selected then
all

            '
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT'
            + --count number of tables requested (number of commas plus one)

            '
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')'
            + --Count of the tables requested how many exist in the db

            '
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'','')) '
            + --only if the count matches (all the tables exist in the requested db)
then run the script

            '
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
                Insert  [#SorDetail]
                ( [DatabaseName]
                , [SalesOrder]
                , [SalesOrderLine]
                , [LineType]
                , [MStockCode]
                , [MStockDes]
                , [MOrderQty]
                , [MOrderUom]
                , [MPrice]
                , [NComment]
                , [NMscProductCls]
                , [NMscChargeValue]
                )
                Select
                    [DatabaseName] = @DBCode
                  , [sd].[SalesOrder]
                  , [sd].[SalesOrderLine]
                  , [sd].[LineType]
                  , [sd].[MStockCode]
                  , [sd].[MStockDes]
                  , [sd].[MOrderQty]
                  , [sd].[MOrderUom]
                  , [sd].[MPrice]
                  , [sd].[NComment]
                  , [sd].[NMscProductCls]
                  , [sd].[NMscChargeValue]
                From
                    [SorDetail] As [sd];
            End
    End';
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-SalesOrders

```
        Declare @SQL3 Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN'
            + --only companies selected in main run, or if companies selected then
all


            '
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT'
            + --count number of tables requested (number of commas plus one)

            '
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')'
            + --Count of the tables requested how many exist in the db

            '
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'','')) '
            + --only if the count matches (all the tables exist in the requested db)
then run the script

            '
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
                Insert  [#SalSalesperson]
                ( [DatabaseName]
                , [Name]
                , [Branch]
                , [Salesperson]
                )
                Select
                    [DatabaseName] = @DBCode
                  , [ss].[Name]
                  , [ss].[Branch]
                  , [ss].[Salesperson]
                From
                    [SalSalesperson] As [ss];
            End
    End';
--Enable this function to check script changes (try to run script directly against
db manually)

--Print @SQL



--execute script against each db, populating the base tables

        Exec [Process].[ExecForEachDB] @cmd = @SQL1;
        Exec [Process].[ExecForEachDB] @cmd = @SQL2;
```

Author: Johnson, Chris

```
        Exec [Process].[ExecForEachDB] @cmd = @SQL3;

--define the results you want to return

        Create Table [#Results]
            (
              [DatabaseName] Varchar(150)            Collate Latin1_General_BIN
            , [SalesOrder] Varchar(35)               Collate Latin1_General_BIN
            , [OrderStatusDescription] Varchar(255) collate latin1_general_bin
            , [CancelledFlag] Char(1)                collate latin1_general_bin
            , [Customer] Varchar(35)                Collate Latin1_General_BIN
            , [CustomerName] Varchar(255)            Collate Latin1_General_BIN
            , [Name] Varchar(150)                    Collate Latin1_General_BIN
            , [CustomerPoNumber] Varchar(150)        Collate Latin1_General_BIN
            , [OrderDate] DateTime2
            , [EntrySystemDate] DateTime2
            , [ReqShipDate] DateTime2
            , [Currency] Varchar(5)                    Collate Latin1_General_BIN
            , [SalesOrderLine] Int
            , [LineTypeDescription] Varchar(150)    Collate Latin1_General_BIN
            , [MStockCode] Varchar(35)               Collate Latin1_General_BIN
            , [MStockDes] Varchar(150)               Collate Latin1_General_BIN
            , [MOrderQty] Numeric(20 , 7)
            , [MOrderUom] Varchar(10)                Collate Latin1_General_BIN
            , [MPrice] Numeric(20 , 3)
            , [NComment] Varchar(100)                Collate Latin1_General_BIN
            , [NMscProductCls] Varchar(20)           Collate Latin1_General_BIN
            , [NMscChargeValue] Numeric(20 , 3)
            );

--Placeholder to create indexes as required


--script to combine base data and insert into results table

        Insert  [#Results]
                ( [DatabaseName]
                , [SalesOrder]
                , [OrderStatusDescription]
                , [CancelledFlag]
                , [Customer]
                , [CustomerName]
                , [Name]
                , [CustomerPoNumber]
                , [OrderDate]
                , [EntrySystemDate]
                , [ReqShipDate]
                , [Currency]
                , [SalesOrderLine]
                , [LineTypeDescription]
                , [MStockCode]
                , [MStockDes]
                , [MOrderQty]
                , [MOrderUom]
                , [MPrice]
                , [NComment]
                , [NMscProductCls]
```

```
                , [NMscChargeValue]
                )
                Select  [sm].[DatabaseName]
                      , [sm].[SalesOrder]
                      , [sos].[OrderStatusDescription]
                      , [sm].[CancelledFlag]
                      , [sm].[Customer]
                      , [sm].[CustomerName]
                      , [ss].[Name]
                      , [sm].[CustomerPoNumber]
                      , [sm].[OrderDate]
                      , [sm].[EntrySystemDate]
                      , [sm].[ReqShipDate]
                      , [sm].[Currency]
                      , [sd].[SalesOrderLine]
                      , [solt].[LineTypeDescription]
                      , [sd].[MStockCode]
                      , [sd].[MStockDes]
                      , [sd].[MOrderQty]
                      , [sd].[MOrderUom]
                      , [sd].[MPrice]
                      , [sd].[NComment]
                      , [sd].[NMscProductCls]
                      , [sd].[NMscChargeValue]
                From    [#SorMaster] As [sm]
                        Left Join [#SorDetail] As [sd] On [sd].[SalesOrder] =
[sm].[SalesOrder]
                                                          And [sd].[DatabaseName] =
[sm].[DatabaseName]
                        Left Join [#SalSalesperson] As [ss] On [ss].[Branch] =
[sm].[Branch]
                                                              And [ss].[Salesperson]
= [sm].[Salesperson]
                                                              And [ss].[Database-
Name] = [sd].[DatabaseName]
                        Left Join [BlackBox].[Lookups].[SalesOrderStatus] As [sos]
On [sm].[OrderStatus] = [sos].[OrderStatusCode]
                                                              And [sos].[Company] =
[sm].[DatabaseName]
                        Left Join [BlackBox].[Lookups].[SalesOrderLineType] As
[solt] On [sd].[LineType] = [solt].[LineTypeCode]
                                                              And [solt].[Company] =
[sm].[DatabaseName];

--return results

        Select  [Company] = [DatabaseName]
              , [SalesOrder]
              , [OrderStatus] = [OrderStatusDescription]
              , [CancelledFlag] = Case When [CancelledFlag] = '' Then 'N'
                                       Else [CancelledFlag]
                                  End
              , [Customer]
              , [CustomerName]
              , [SalesPerons] = [Name]
              , [CustomerPoNumber]
              , [OrderDate] = Cast([OrderDate] As Date)
              , [EntrySystemDate] = Cast([EntrySystemDate] As Date)
              , [ReqShipDate] = Cast([ReqShipDate] As Date)
```

Author: Johnson, Chris

```
                , [Currency]
                , [Line] = [SalesOrderLine]
                , [LineType] = [LineTypeDescription]
                , [StockCode] = [MStockCode]
                , [StockDescription] = [MStockDes]
                , [OrderQty] = [MOrderQty]
                , [OrderUom] = [MOrderUom]
                , [Price] = [MPrice]
                , [Comment] = [NComment]
                , [ProductClass] = [NMscProductCls]
                , [ChargeValue] = [NMscChargeValue]
        From    [#Results];

    End;

GO
EXEC sp_addextendedproperty N'MS_Description', N'list of sales orders ', 'SCHEMA',
N'Report', 'PROCEDURE', N'UspResults_SalesOrders', NULL, NULL
GO
```

Uses

[Lookups].[SalesOrderLineType]
[Lookups].[SalesOrderStatus]
[Process].[ExecForEachDB]
[Process].[UspInsert_RedTagLogs]
[Report]

[Report].[UspResults_SalesOrderStats]

MS_Description

data for sales order stats report

Parameters

Name	Data Type	Max Length (Bytes)
@Company	varchar(max)	max
@RedTagType	char	1
@RedTagUse	varchar(500)	500

SQL Script

```
CREATE Proc [Report].[UspResults_SalesOrderStats]
    (
      @Company Varchar(Max)
    , @RedTagType Char(1)
    , @RedTagUse Varchar(500)
    )
As
    Begin

        Set NoCount On;

        If IsNumeric(@Company) = 0
            Begin
                Select  @Company = Upper(@Company);
            End;

--Red tag

        Declare @RedTagDB Varchar(255)= Db_Name();
        Exec [Process].[UspInsert_RedTagLogs] @StoredProcDb = 'BlackBox' ,
            @StoredProcSchema = 'Report' ,
            @StoredProcName = 'UspResults_SalesOrderStats' ,
            @UsedByType = @RedTagType , @UsedByName = @RedTagUse ,
            @UsedByDb = @RedTagDB;

--list the tables that are to be pulled back from each DB - if they are not found
the script will not be run against that db

        Declare @ListOfTables Varchar(Max) = 'ArCustomer,SorMaster,SorDetail,Gen-
JournalDetail,CusSorMaster+';

--create temporary tables to be pulled from different databases, including a column
to id

        Create Table [#ArCustomer]
            (
              [DatabaseName] Varchar(150)        collate Latin1_General_BIN
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-SalesOrderStats

```
            , [Customer] Varchar(15)            collate Latin1_General_BIN
            , [Name] Varchar(50)                collate Latin1_General_BIN
            );
        Create Table [#SorMaster]
            (
              [DatabaseName] Varchar(150)        collate Latin1_General_BIN
            , [CustomerPoNumber] Varchar(30)    collate Latin1_General_BIN
            , [EntrySystemDate] DateTime
            , [Currency] Char(3)                collate Latin1_General_BIN
            , [SalesOrder] Varchar(20)            collate Latin1_General_BIN
            , [Customer] Varchar(15)            collate Latin1_General_BIN
            , [DocumentType] Varchar(10)        collate Latin1_General_BIN
            , [LastInvoice] Varchar(20)            collate Latin1_General_BIN
            , [ShipAddress5] Varchar(40)        collate Latin1_General_BIN
            );
        Create Table [#SorDetail]
            (
              [DatabaseName] Varchar(150)        collate Latin1_General_BIN
            , [MStockCode] Varchar(30)            collate Latin1_General_BIN
            , [MStockDes] Varchar(50)           collate Latin1_General_BIN
            , [MOrderQty] Numeric(20 , 8)
            , [MPrice] Numeric(20 , 8)
            , [MLineShipDate] DateTime
            , [SalesOrderLine] Int
            , [SalesOrder] Varchar(20)            collate Latin1_General_BIN
            );
        Create Table [#CusSorMasterPlus]
            (
              [DatabaseName] Varchar(150)        collate Latin1_General_BIN
            , [AttentionOf] Varchar(60)            collate Latin1_General_BIN
            , [AcceptedDate] DateTime
            , [SalesOrder] Varchar(20)            collate Latin1_General_BIN
            );
        Create Table [#GenJournalDetail]
            (
              [DatabaseName] Varchar(150)        collate Latin1_General_BIN
            , [SubModArInvoice] Varchar(20)        collate Latin1_General_BIN
            , [Reference] Varchar(50)           collate Latin1_General_BIN
            , [EntryDate] Date
            );



--create script to pull data from each db into the tables

        Declare @SQLArCustomer Varchar(Max) = 'USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            BEGIN
                Insert [#ArCustomer]
                        ( [DatabaseName]
                        , [Customer]
```

Author: Johnson, Chris

```
                        , [Name]
                        )
                SELECT [DatabaseName]=@DBCode
                     , [AC].[Customer]
                     , [AC].[Name] FROM [ArCustomer] [AC]
            End
    End';
        Declare @SQLSorMaster Varchar(Max) = 'USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            BEGIN
                Insert [#SorMaster]
                        ( [DatabaseName]
                        , [CustomerPoNumber]
                        , [EntrySystemDate]
                        , [Currency]
                        , [SalesOrder]
                        , [Customer]
                        , [DocumentType]
                        , [LastInvoice]
                        , [ShipAddress5]
                        )
                SELECT [DatabaseName]=@DBCode
                     , [SM].[CustomerPoNumber]
                     , [SM].[EntrySystemDate]
                     , [SM].[Currency]
                     , [SM].[SalesOrder]
                     , [SM].[Customer]
                     , [SM].[DocumentType]
                     , [LastInvoice]
                     , [ShipAddress5]
                FROM [SorMaster] [SM]
            End
    End';
        Declare @SQLSorDetail Varchar(Max) = 'USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            BEGIN
                Insert [#SorDetail]
                        ( [DatabaseName]
                        , [MStockCode]
                        , [MStockDes]
                        , [MOrderQty]
                        , [MPrice]
                        , [MLineShipDate]
                        , [SalesOrderLine]
                        , [SalesOrder]
```

Author: Johnson, Chris

```
                        )
                SELECT [DatabaseName]=@DBCode
                     , [SD].[MStockCode]
                     , [SD].[MStockDes]
                     , [SD].[MOrderQty]
                     , [SD].[MPrice]
                     , [SD].[MLineShipDate]
                     , [SD].[SalesOrderLine]
                     , [SD].[SalesOrder] FROM [SorDetail] [SD]
            End
    End';
        Declare @SQLCusSorMasterPlus Varchar(Max) = 'USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            BEGIN
                Insert [#CusSorMasterPlus]
                        ( [DatabaseName]
                        , [AttentionOf]
                        , [AcceptedDate]
                        , [SalesOrder]
                        )
                SELECT [DatabaseName]=@DBCode
                     , [CSMP].[AttentionOf]
                     , [CSMP].[AcceptedDate]
                     , [CSMP].[SalesOrder] FROM [CusSorMaster+] [CSMP]
            End
    End';
        Declare @SQLGenJournalDetail Varchar(Max) = 'USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            BEGIN
            Insert [#GenJournalDetail]
                    ( [DatabaseName]
                    , [SubModArInvoice]
                    , [Reference]
                    , [EntryDate]
                    )
            Select Distinct
                    [DatabaseName]=@DBCode
                    , [SubModArInvoice]
                    , [Reference]
                    , [EntryDate]
            From    [dbo].[GenJournalDetail]
            Where   [Reference] = ''Invoice''
            And [SubModArInvoice] <> ''''
            End
    End';
--Enable this function to check script changes (try to run script directly against
```

Author: Johnson, Chris

```
db manually)

--Print @SQL


--execute script against each db, populating the base tables

        Exec [Process].[ExecForEachDB_WithTableCheck] @cmd = @SQLArCustomer ,
            @SchemaTablesToCheck = @ListOfTables;
        Exec [Process].[ExecForEachDB_WithTableCheck] @cmd = @SQLSorMaster ,
            @SchemaTablesToCheck = @ListOfTables;
        Exec [Process].[ExecForEachDB_WithTableCheck] @cmd = @SQLSorDetail ,
            @SchemaTablesToCheck = @ListOfTables;
        Exec [Process].[ExecForEachDB_WithTableCheck] @cmd = @SQLCusSorMasterPlus ,
            @SchemaTablesToCheck = @ListOfTables;
        Exec [Process].[ExecForEachDB_WithTableCheck] @cmd = @SQLGenJournalDetail ,
            @SchemaTablesToCheck = @ListOfTables;

--define the results you want to return

        Create Table [#Results]
            (
              [DatabaseName] Varchar(150)        collate Latin1_General_BIN
            , [Customer] Varchar(15)            collate Latin1_General_BIN
            , [CustomerName] Varchar(50)        collate Latin1_General_BIN
            , [CustomerPoNumber] Varchar(30)    collate Latin1_General_BIN
            , [Contact] Varchar(60)               collate Latin1_General_BIN
            , [EntrySystemDate] DateTime
            , [AcceptedDate] DateTime
            , [StockCode] Varchar(30)            collate Latin1_General_BIN
            , [StockDescription] Varchar(50)    collate Latin1_General_BIN
            , [OrderQty] Numeric(20 , 8)
            , [Price] Numeric(20 , 8)
            , [Currency] Varchar(10)            collate Latin1_General_BIN
            , [ShipDate] DateTime
            , [SalesOrder] Varchar(20)            collate Latin1_General_BIN
            , [SOLine] Int
            , [DocumentType] Varchar(250)        collate Latin1_General_BIN
            , [LastInvoice] Varchar(20)           collate Latin1_General_BIN
            , [ProFormaDate] DateTime
            , [Country] Varchar(40)               collate Latin1_General_BIN
            , [InvoiceEntryDate] Date
            );

--Placeholder to create indexes as required

        Create Table [#ProformaDates]
            (
              [DatabaseName] Varchar(150)        collate Latin1_General_BIN
            , [ProFormaDate] DateTime2
            , [SalesOrder] Varchar(20)            collate Latin1_General_BIN
            );

        Insert  [#ProformaDates]
                ( [DatabaseName]
                , [ProFormaDate]
                , [SalesOrder]
```

Author: Johnson, Chris

```
                )
                Select  [DatabaseName] = Replace(Upper([SM2].[DatabaseName]) ,
                                                 'SYSPROCOMPANY' , '')
                      , [ProFormaDate] = Max([SM2].[SignatureDateTime])
                      , [SM2].[SALESORDER]
                From    [History].[SorMaster] [SM2]
                        Inner Join [Lookups].[ProformaDocTypes] [PDT]
                            On [PDT].[DOCUMENTFORMAT] = [SM2].[DOCUMENTFORMAT]
                               And [PDT].[DOCUMENTTYPE] = [SM2].[DOCUMENTTYPE]
                Group By Replace(Upper([SM2].[DatabaseName]) , 'SYSPROCOMPANY' ,
                                 '')
                      , [SM2].[SALESORDER];

--script to combine base data and insert into results table

        Insert  [#Results]
                ( [DatabaseName]
                , [Customer]
                , [CustomerName]
                , [CustomerPoNumber]
                , [Contact]
                , [EntrySystemDate]
                , [AcceptedDate]
                , [StockCode]
                , [StockDescription]
                , [OrderQty]
                , [Price]
                , [Currency]
                , [ShipDate]
                , [SalesOrder]
                , [SOLine]
                , [DocumentType]
                , [LastInvoice]
                , [ProFormaDate]
                , [Country]
                , [InvoiceEntryDate]
                )
                Select  [AC].[DatabaseName]
                      , [AC].[Customer]
                      , [AC].[Name]
                      , [SM].[CustomerPoNumber]
                      , [Contact] = [CSM].[AttentionOf]
                      , [SM].[EntrySystemDate]
                      , [CSM].[AcceptedDate]
                      , [StockCode] = [SD].[MStockCode]
                      , [StockDescription] = [SD].[MStockDes]
                      , [OrderQty] = [SD].[MOrderQty]
                      , [Price] = [SD].[MPrice]
                      , [SM].[Currency]
                      , [ShipDate] = [SD].[MLineShipDate]
                      , [SalesOrder] = Case When IsNumeric([SM].[SalesOrder]) = 1
                                            Then Convert(Varchar(20) , Convert(Int ,
[SM].[SalesOrder]))
                                            Else [SM].[SalesOrder]
                                       End
                      , [SOLine] = [SD].[SalesOrderLine]
                      , [DocumentType] = [SODT].[DocumentTypeDesc]
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-SalesOrderStats

```
                      , [LastInvoice] = Case When IsNumeric([SM].[LastInvoice]) = 1
                                             Then Convert(Varchar(20) , Convert(Big-
Int , [SM].[LastInvoice]))
                                             Else [SM].[LastInvoice]
                                        End
                      , [PD].[ProFormaDate]
                      , [Country] = Case When [SM].[ShipAddress5] = ''
                                         Then Null
                                         Else [SM].[ShipAddress5]
                                    End
                      , [GJD].[EntryDate]
                From    [#ArCustomer] [AC]
                        Inner Join [#SorMaster] [SM]
                            On [SM].[Customer] = [AC].[Customer]
                        Inner Join [#SorDetail] [SD]
                            On [SD].[SalesOrder] = [SM].[SalesOrder]
                        Left Join [#CusSorMasterPlus] [CSM]
                            On [CSM].[SalesOrder] = [SM].[SalesOrder]
                        Left Join [BlackBox].[Lookups].[SalesOrderDocumentType]
[SODT]
                            On [SODT].[DocumentType] = [SM].[DocumentType]
                        Left Join [#ProformaDates] [PD]
                            On [PD].[DatabaseName] = [SM].[DatabaseName]
                               And [PD].[SalesOrder] = [SM].[SalesOrder]
                        Left Join [#GenJournalDetail] [GJD]
                            On [SM].[LastInvoice] = [GJD].[SubModArInvoice]
                               And [GJD].[DatabaseName] = [SM].[DatabaseName]
                Order By [SalesOrder] Desc;

        Set NoCount Off;
--return results

        Select  [R].[DatabaseName]
              , [R].[Customer]
              , [R].[CustomerName]
              , [R].[CustomerPoNumber]
              , [R].[Contact]
              , [R].[EntrySystemDate]
              , [R].[AcceptedDate]
              , [R].[StockCode]
              , [R].[StockDescription]
              , [R].[OrderQty]
              , [R].[Price]
              , [R].[Currency]
              , [R].[ShipDate]
              , [R].[SalesOrder]
              , [R].[SOLine]
              , [R].[DocumentType]
              , [R].[LastInvoice]
              , [CN].[CompanyName]
              , [CN].[ShortName]
              , [CompanyCurrency] = [CN].[Currency]
              , [R].[ProFormaDate]
              , [R].[Country]
              , [R].[InvoiceEntryDate]
        From    [#Results] [R]
                Left Join [Lookups].[CompanyNames] [CN]
```

Author: Johnson, Chris

```
                    On [R].[DatabaseName] = [CN].[Company];


    End;


GO
EXEC sp_addextendedproperty N'MS_Description', N'data for sales order stats report',
'SCHEMA', N'Report', 'PROCEDURE', N'UspResults_SalesOrderStats', NULL, NULL
GO
```

Uses

[Lookups].[CompanyNames]
[Lookups].[ProformaDocTypes]
[Lookups].[SalesOrderDocumentType]
[Process].[ExecForEachDB_WithTableCheck]
[Process].[UspInsert_RedTagLogs]
[History].[SorMaster]
[Report]

[Report].[UspResults_ScrapCosts]

MS_Description

list of scrap costs

Parameters

Name	Data Type	Max Length (Bytes)
@Company	varchar(max)	max
@RedTagType	char	1
@RedTagUse	varchar(500)	500

SQL Script

```
CREATE Proc [Report].[UspResults_ScrapCosts]
    (
      @Company Varchar(Max)
    , @RedTagType Char(1)
    , @RedTagUse Varchar(500)
    )
As
    Begin
/*
Template designed by Chris Johnson, Prometic Group September 2015
Stored procedure set out to query multiple databases with the same information and
return it in a collated format
--exec [Report].[UspResults_ScrapCosts] '10'
*/
        If IsNumeric(@Company) = 0
            Begin
                Select  @Company = Upper(@Company);
            End;

--remove nocount on to speed up query

        Set NoCount On;
--Red tag

        Declare @RedTagDB Varchar(255)= Db_Name();
        Exec [Process].[UspInsert_RedTagLogs] @StoredProcDb = 'BlackBox' ,
            @StoredProcSchema = 'Report' ,
            @StoredProcName = 'UspResults_ScrapCosts' ,
            @UsedByType = @RedTagType , @UsedByName = @RedTagUse ,
            @UsedByDb = @RedTagDB;
--list the tables that are to be pulled back from each DB - if they are not found
the script will not be run against that db

        Declare @ListOfTables Varchar(Max) = 'InvInspect,InvInspectDet';

--create temporary tables to be pulled from different databases, including a column
to id
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-ScrapCosts

```
        Create Table [#InvInspect]
            (
              [DatabaseName] Varchar(150)    Collate Latin1_General_BIN
            , [Lot] Varchar(50)                Collate Latin1_General_BIN
            , [QtyScrapped] Numeric(20 , 8)
            , [StockCode] Varchar(50)        collate latin1_general_bin
            , [PoPrice] Numeric(20 , 8)
            , [PriceUom] Varchar(10)        collate latin1_general_bin
            , [PrcFactor] Char(1)            collate latin1_general_bin
            , [ConvFactPrcUom] Int
            , [ConvFactOrdUom] Int
            , [Grn] Varchar(50)                Collate Latin1_General_BIN
            );
        Create Table [#InvInspectDet]
            (
              [DatabaseName] Varchar(150)    Collate Latin1_General_BIN
            , [TrnQty] Numeric(20 , 8)
            , [TrnDate] Date
            , [RejectScrapCode] Varchar(50)    collate latin1_general_bin
            , [TrnType] Char(1)
            , [Grn] Varchar(50)                Collate Latin1_General_BIN
            , [Lot] Varchar(50)                Collate Latin1_General_BIN
            );

--create script to pull data from each db into the tables

        Declare @SQL1 Varchar(Max) = 'USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN'
            + --only companies selected in main run, or if companies selected then
all

            '
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT'
            + --count number of tables requested (number of commas plus one)

            '
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')'
            + --Count of the tables requested how many exist in the db

            '
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'','')) '
            + --only if the count matches (all the tables exist in the requested db)
then run the script

            '
            If @ActualCountOfTables=@RequiredCountOfTables
```

Author: Johnson, Chris

```
            BEGIN
            Insert [#InvInspect]
                    ( [DatabaseName]
                    , [Lot]
                    , [QtyScrapped]
                    , [StockCode]
                    , [PoPrice]
                    , [PriceUom]
                    , [PrcFactor]
                    , [ConvFactPrcUom]
                    , [ConvFactOrdUom]
                    , [Grn]
                    )
            SELECT [DatabaseName]=@DBCode
                 , [ii].[Lot]
                 , [ii].[QtyScrapped]
                 , [ii].[StockCode]
                 , [ii].[PoPrice]
                 , [ii].[PriceUom]
                 , [ii].[PrcFactor]
                 , [ii].[ConvFactPrcUom]
                 , [ii].[ConvFactOrdUom]
                 , [Grn]
            From [InvInspect] As [ii]
            where [ii].[QtyScrapped]<>0
            End
    End';
        Declare @SQL2 Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN'
            + --only companies selected in main run, or if companies selected then
all

            '
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT'
            + --count number of tables requested (number of commas plus one)

            '
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')'
            + --Count of the tables requested how many exist in the db

            '
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'',''))  '
            + --only if the count matches (all the tables exist in the requested db)
then run the script

            '
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-ScrapCosts

```
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
            Insert [#InvInspectDet]
                    ( [DatabaseName]
                    , [TrnQty]
                    , [TrnDate]
                    , [RejectScrapCode]
                    , [TrnType]
                    ,[Grn]
                    ,[Lot]
                    )
            SELECT [DatabaseName]=@DBCode
                 , [iid].[TrnQty]
                 , [iid].[TrnDate]
                 , [iid].[RejectScrapCode]
                 , [iid].[TrnType]
                 ,[Grn]
                 ,[Lot]
            From [InvInspectDet] As [iid]
            where [iid].[TrnType]=''S''
            End
    End';

--Enable this function to check script changes (try to run script directly against
db manually)

--Print @SQL


--execute script against each db, populating the base tables

        Exec [Process].[ExecForEachDB] @cmd = @SQL1;
        Exec [Process].[ExecForEachDB] @cmd = @SQL2;

--define the results you want to return

        Create Table [#Results]
            (
              [DatabaseName] Varchar(150)        Collate Latin1_General_BIN
            , [Lot] Varchar(35)                    Collate Latin1_General_BIN
            , [TotalScrapped] Numeric(20 , 8)
            , [TrnQty] Numeric(20 , 8)
            , [StockCode] Varchar(50)            collate latin1_general_bin
            , [PoPrice] Numeric(20 , 8)
            , [TrnDate] Date
            , [PriceUom] Varchar(10)            collate latin1_general_bin
            , [PrcFactor] Char(1)                collate latin1_general_bin
            , [TotalValue] Numeric(20 , 8)
            , [Value] Numeric(20 , 8)
            , [RejectScrapCode] Varchar(50)        collate latin1_general_bin
            , [ConvFactPrcUom] Int
            , [ConvFactOrdUom] Int
            );

--Placeholder to create indexes as required
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-ScrapCosts

```
--script to combine base data and insert into results table

        Insert  [#Results]
                ( [DatabaseName]
                , [Lot]
                , [TotalScrapped]
                , [TrnQty]
                , [StockCode]
                , [PoPrice]
                , [TrnDate]
                , [PriceUom]
                , [PrcFactor]
                , [TotalValue]
                , [Value]
                , [RejectScrapCode]
                , [ConvFactPrcUom]
                , [ConvFactOrdUom]
                )
                Select  [ii].[DatabaseName]
                      , [ii].[Lot]
                      , [TotalScrapped] = [ii].[QtyScrapped]
                      , [iid].[TrnQty]
                      , [ii].[StockCode]
                      , [ii].[PoPrice]
                      , [iid].[TrnDate]
                      , [ii].[PriceUom]
                      , [ii].[PrcFactor]
                      , [TotalValue] = [ii].[PoPrice] * [ii].[QtyScrapped]
                      , [Value] = [ii].[PoPrice] * [iid].[TrnQty]
                      , [iid].[RejectScrapCode]
                      , [ii].[ConvFactPrcUom]
                      , [ii].[ConvFactOrdUom]
                From    [#InvInspect] As [ii]
                        Inner Join [#InvInspectDet] As [iid] On [iid].[Grn] =
[ii].[Grn]
                                                                And [iid].[Lot] =
[ii].[Lot]
                                                                And [iid].[Database-
Name] = [ii].[DatabaseName];



--return results

        Select  [cn].[CompanyName]
              , [r].[Lot]
              , [r].[TotalScrapped]
              , [r].[TrnQty]
              , [r].[StockCode]
              , [r].[PoPrice]
              , [r].[TrnDate]
              , [r].[PriceUom]
              , [r].[PrcFactor]
              , [r].[TotalValue]
              , [r].[Value]
              , [r].[RejectScrapCode]
              , [r].[ConvFactPrcUom]
              , [r].[ConvFactOrdUom]
```

Author: Johnson, Chris

```
        From    [#Results] As [r]
                Left Join [Lookups].[CompanyNames] As [cn] On [r].[DatabaseName] =
[cn].[Company];


    End;


GO
EXEC sp_addextendedproperty N'MS_Description', N'list of scrap costs', 'SCHEMA',
N'Report', 'PROCEDURE', N'UspResults_ScrapCosts', NULL, NULL
GO
```

Uses

[Lookups].[CompanyNames]
[Process].[ExecForEachDB]
[Process].[UspInsert_RedTagLogs]
[Report]

Author: Johnson, Chris

[Report].[UspResults_SearchForMissingIndexes]

MS_Description

proc to evaluate any missing indexes that can be added

Parameters

Name	Data Type	Max Length (Bytes)
@DbSearch	varchar(300)	300

SQL Script

```
CREATE Proc [Report].[UspResults_SearchForMissingIndexes] ( @DbSearch Varchar(300) )
As /*
Stored Procedure created by Chris Johnson
3rd February 2016

The purpose of this stored procedure is to search for missing indexes
Based off work here - http://blog.sqlauthority.com/2011/01/03/sql-server-2008-missing-index-script-download/
*/
--Cater for nulls and append %

    Set @DbSearch = '%' + Coalesce(@DbSearch , '') + '%';

    Declare @SQLScript Varchar(Max);

    Create Table [#ResultSets]
        (
          [DatabaseName] Varchar(300)    collate latin1_general_bin
        , [AvgEstimatedImpact] Float
        , [LastUserSeek] DateTime2
        , [UserScans] BigInt
        , [UserSeeks] BigInt
        , [AvgUserImpact] Float
        , [TableName] Varchar(500)        collate latin1_general_bin
        , [Create_Statement] Varchar(Max)
        );




    Set @SQLScript = 'Insert [#ResultSets]
                ( [DatabaseName]
                , [AvgEstimatedImpact]
                , [LastUserSeek]
                , [UserScans]
                , [UserSeeks]
                , [AvgUserImpact]
```

Author: Johnson, Chris

```
                , [TableName]
                , [Create_Statement]
                )
                Select  [DatabaseName] = [d].[name]
      , [AvgEstimatedImpact] = [dm_migs].[avg_user_impact]
                          * ( [dm_migs].[user_seeks] + [dm_migs].[user_scans] )
      , [LastUserSeek]        = [dm_migs].[last_user_seek]
      , [UserScans]            = [dm_migs].[user_scans]
      , [UserSeeks]            = [dm_migs].[user_seeks]
      , [AvgUserImpact]        = [dm_migs].[avg_user_impact]
      , [TableName]            = Object_Name([dm_mid].[object_id] ,
                                    [dm_mid].[database_id])
      , [Create_Statement] = ''CREATE INDEX '' + Replace(QuoteName(''IX_''
                                                                   + Object_-
Name([dm_mid].[object_id] ,

[dm_mid].[database_id])
                                                                   + ''_''
                                                                   +
Replace(Replace(Replace(Replace(Coalesce([dm_mid].[equality_columns]
                                                                   + '', '' , '''')
                                                                   +
Coalesce([dm_mid].[inequality_columns] ,
                                                                   '''') , ''], ['' ,
                                                                   ''_'') , ''['' , '''')
,
                                                                   '']'' , '''') , '',''
,
                                                                   '''')) , '' '' , '''')
        + '' ON '' + [dm_mid].[statement] + '' (''
        + Replace(Replace(Coalesce([dm_mid].[equality_columns] + ''_'' , '''')
                          + Coalesce([dm_mid].[inequality_columns] , '''') + '')'' ,
                          ''],['' , ''],['') , ''_'' , '''') + Coalesce('' INCLUDE
(''
                                                                   +
[dm_mid].[included_columns]
                                                                   + '')'' , '''')
From    [sys].[dm_db_missing_index_groups] [dm_mig]
        Inner Join [sys].[dm_db_missing_index_group_stats] [dm_migs] On
[dm_migs].[group_handle] = [dm_mig].[index_group_handle]
        Inner Join [sys].[dm_db_missing_index_details] [dm_mid] On
[dm_mig].[index_handle] = [dm_mid].[index_handle]
        Inner Join [sys].[databases] [d] On [d].[database_id] =
[dm_mid].[database_id]

        Where   Lower([d].[name]) Like Lower(''' + @DbSearch + ''');';

    Exec (@SQLScript)


    Select  [RS].[DatabaseName]
          , [RS].[AvgEstimatedImpact]
          , [RS].[LastUserSeek]
          , [RS].[UserScans]
          , [RS].[UserSeeks]
          , [RS].[AvgUserImpact]
          , [RS].[TableName]
          , [RS].[Create_Statement]
    From    [#ResultSets] As [RS]
    Where [RS].[TableName] Is Not Null
```

Author: Johnson, Chris

```
    Order By [RS].[AvgEstimatedImpact] Desc;
GO
EXEC sp_addextendedproperty N'MS_Description', N'proc to evaluate any missing
indexes that can be added', 'SCHEMA', N'Report', 'PROCEDURE', N'UspResults_SearchFor-
MissingIndexes', NULL, NULL
GO
```

Uses

[Report]

[Report].[UspResults_StockDispatches]

MS_Description

details of stock dispatched

Parameters

Name	Data Type	Max Length (Bytes)
@Company	varchar(max)	max
@RedTagType	char	1
@RedTagUse	varchar(500)	500

SQL Script

```
CREATE Proc [Report].[UspResults_StockDispatches]
    (
      @Company Varchar(Max)
    , @RedTagType Char(1)
    , @RedTagUse Varchar(500)
    )
As
    Begin
/*
Template designed by Chris Johnson, Prometic Group March 2016
Stored procedure set out to query multiple databases with the same information and
return it in a collated format
*/
        If IsNumeric(@Company) = 0
            Begin
                Select  @Company = Upper(@Company);
            End;

--remove nocount on to speed up query

        Set NoCount On;

--Red tag

        Declare @RedTagDB Varchar(255)= Db_Name();
        Exec [Process].[UspInsert_RedTagLogs] @StoredProcDb = 'BlackBox' ,
            @StoredProcSchema = 'Report' ,
            @StoredProcName = 'UspResults_StockDispatches' ,
            @UsedByType = @RedTagType , @UsedByName = @RedTagUse ,
            @UsedByDb = @RedTagDB;

--list the tables that are to be pulled back from each DB - if they are not found
the script will not be run against that db

        Declare @ListOfTables Varchar(Max) = 'AssetDepreciation,TblApTerms';

--create temporary tables to be pulled from different databases, including a column
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-StockDispatches

```
to id

        Create Table [#LotTransactionsDispatches]
            (
              [DatabaseName] Varchar(150)        collate latin1_general_bin
            , [Lot] Varchar(50)                    collate latin1_general_bin
            , [StockCode] Varchar(30)            collate latin1_general_bin
            , [Customer] Varchar(15)            collate latin1_general_bin
            , [SalesOrder] Varchar(20)             collate latin1_general_bin
            , [SalesOrderLine] Int
            , [TrnQuantity] Numeric(20 , 8)
            , [TrnValue] Numeric(20 , 2)
            , [TrnType] Char(1)                    collate latin1_general_bin
            , [TrnDate] Date
            , [OldExpiryDate] Date
            , [NewExpiryDate] Date
            , [Job] Varchar(20)                    collate latin1_general_bin
            , [Bin] Varchar(20)                    collate latin1_general_bin
            , [UnitCost] Numeric(20 , 2)
            , [Narration] Varchar(100)             collate latin1_general_bin
            , [Reference] Varchar(30)            collate latin1_general_bin
            , [Warehouse] Varchar(10)            collate latin1_general_bin
            , [JobPurchOrder] Varchar(20)        collate latin1_general_bin
            );
        Create Table [#InvMasterDispatches]
            (
              [DatabaseName] Varchar(150)        collate latin1_general_bin
            , [StockCode] Varchar(30)            collate latin1_general_bin
            , [Description] Varchar(50)            collate latin1_general_bin
            , [StockUom] Varchar(10)            collate latin1_general_bin
            );
        Create Table [#ArCustomerDispatches]
            (
              [DatabaseName] Varchar(150)        collate latin1_general_bin
            , [Customer] Varchar(15)            collate latin1_general_bin
            , [Name] Varchar(50)                collate latin1_general_bin
            );
        Create Table [#WipMasterDispatches]
            (
              [DatabaseName] Varchar(150)        collate latin1_general_bin
            , [Job] Varchar(20)                    collate latin1_general_bin
            , [JobDescription] Varchar(50)         collate latin1_general_bin
            , [JobClassification] Varchar(10)    collate latin1_general_bin
            , [SellingPrice] Numeric(20 , 2)
            );
        Create Table [#SorMasterDispatches]
            (
              [DatabaseName] Varchar(150)        collate latin1_general_bin
            , [SalesOrder] Varchar(20)             collate latin1_general_bin
            , [CustomerPoNumber] Varchar(30)    collate latin1_general_bin
            );
        Create Table [#SorDetailDispatches]
            (
              [DatabaseName] Varchar(150)        collate latin1_general_bin
            , [SalesOrder] Varchar(20)             collate latin1_general_bin
            , [SalesOrderLine] Int
            , [MPrice] Numeric(20 , 2)
```

Author: Johnson, Chris

```
            );

--create script to pull data from each db into the tables

        Declare @SQLLotTransactions Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables
            + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'',''))
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
                Insert [#LotTransactionsDispatches]
                        ( [DatabaseName]
                        , [Lot]
                        , [StockCode]
                        , [Customer]
                        , [SalesOrder]
                        , [SalesOrderLine]
                        , [TrnQuantity]
                        , [TrnValue]
                        , [TrnType]
                        , [TrnDate]
                        , [OldExpiryDate]
                        , [NewExpiryDate]
                        , [Job]
                        , [Bin]
                        , [UnitCost]
                        , [Narration]
                        , [Reference]
                        , [Warehouse]
                        , [JobPurchOrder]
                        )
                SELECT [DatabaseName]=@DBCode
                     , [LT].[Lot]
                     , [LT].[StockCode]
                     , [LT].[Customer]
                     , [LT].[SalesOrder]
                     , [LT].[SalesOrderLine]
                     , [LT].[TrnQuantity]
                     , [LT].[TrnValue]
                     , [LT].[TrnType]
                     , [LT].[TrnDate]
                     , [LT].[OldExpiryDate]
                     , [LT].[NewExpiryDate]
                     , [LT].[Job]
```

Author: Johnson, Chris

```
                    , [LT].[Bin]
                    , [LT].[UnitCost]
                    , [LT].[Narration]
                    , [LT].[Reference]
                    , [LT].[Warehouse]
                    , [LT].[JobPurchOrder] FROM [LotTransactions] As [LT]
            End
    End';
        Declare @SQLInvMaster Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables
            + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'',''))
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
                Insert [#InvMasterDispatches]
                        ( [DatabaseName]
                        , [StockCode]
                        , [Description]
                        , [StockUom]
                        )
                SELECT [DatabaseName]=@DBCode
                     , [IM].[StockCode]
                     , [IM].[Description]
                     , [IM].[StockUom] FROM [InvMaster] As [IM]
            End
    End';
        Declare @SQLArCustomer Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables
            + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'',''))
```

Author: Johnson, Chris

```
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
            Insert [#ArCustomerDispatches]
                    ( [DatabaseName]
                    , [Customer]
                    , [Name]
                    )
            SELECT [DatabaseName]=@DBCode
                 , [AC].[Customer]
                 , [AC].[Name] FROM [ArCustomer] As [AC]
            End
    End';
        Declare @SQLWipMaster Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables
            + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'',''))
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
            Insert [#WipMasterDispatches]
                    ( [DatabaseName]
                    , [Job]
                    , [JobDescription]
                    , [JobClassification]
                    , [SellingPrice]
                    )
            SELECT [DatabaseName]=@DBCode
                 , [WM].[Job]
                 , [WM].[JobDescription]
                 , [WM].[JobClassification]
                 , [WM].[SellingPrice] FROM [WipMaster] As [WM]
            End
    End';
        Declare @SQLSorMaster Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables
            + '''
                    , @RequiredCountOfTables INT
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-StockDispatches

```
                    , @ActualCountOfTables INT
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'',''))
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
            Insert [#SorMasterDispatches]
                    ( [DatabaseName]
                    , [SalesOrder]
                    , [CustomerPoNumber]
                    )
            SELECT [DatabaseName]=@DBCode
                 , [SM].[SalesOrder]
                 , [SM].[CustomerPoNumber] FROM [SorMaster] As [SM]
            End
    End';
        Declare @SQLSorDetail Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables
            + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'',''))
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
            Insert [#SorDetailDispatches]
                ( [DatabaseName]
                , [SalesOrder]
                , [SalesOrderLine]
                , [MPrice]
                )
            Select  [DatabaseName]=@DBCode
                  , [SD].[SalesOrder]
                  , [SD].[SalesOrderLine]
                  , [SD].[MPrice]
            From    [SorDetail] As [SD]
            End
    End';
--Enable this function to check script changes (try to run script directly against
db manually)

--Print @SQL



--execute script against each db, populating the base tables
```

Author: Johnson, Chris

```
        Begin
            Exec [Process].[ExecForEachDB] @cmd = @SQLLotTransactions;
            Exec [Process].[ExecForEachDB] @cmd = @SQLInvMaster;
            Exec [Process].[ExecForEachDB] @cmd = @SQLArCustomer;
            Exec [Process].[ExecForEachDB] @cmd = @SQLWipMaster;
            Exec [Process].[ExecForEachDB] @cmd = @SQLSorMaster;
            Exec [Process].[ExecForEachDB] @cmd = @SQLSorDetail;
        End;

--define the results you want to return

        Create Table [#ResultsDispatches]
            (
              [Company] Varchar(150)                    collate latin1_general_bin
            , [CompanyName] Varchar(250)                collate latin1_general_bin
            , [OriginalBatch] Varchar(20)                collate latin1_general_bin
            , [Lot] Varchar(50)                           collate
latin1_general_bin
            , [StockCode] Varchar(30)                    collate latin1_general_bin
            , [Description] Varchar(50)                   collate
latin1_general_bin
            , [Customer] Varchar(15)                    collate latin1_general_bin
            , [CustomerName] Varchar(50)                collate latin1_general_bin
            , [JobDescription] Varchar(50)                collate latin1_general_bin
            , [JobClassification] Varchar(10)            collate latin1_general_bin
            , [SellingPrice] Numeric(20 , 2)
            , [SalesOrder] Varchar(20)                    collate latin1_general_bin
            , [SalesOrderLine] Int
            , [TrnQuantity] Numeric(20 , 8)
            , [TrnValue] Numeric(20 , 2)
            , [TrnType] Char(1)                           collate
latin1_general_bin
            , [AmountModifier] Float
            , [TrnDate] Date
            , [OldExpiryDate] Date
            , [NewExpiryDate] Date
            , [Job] Varchar(20)                           collate
latin1_general_bin
            , [Bin] Varchar(20)                           collate
latin1_general_bin
            , [CustomerPoNumber] Varchar(30)            collate latin1_general_bin
            , [UnitCost] Numeric(20 , 2)
            , [WarehouseDescription] Varchar(200)        collate latin1_general_bin
            , [StockUom] Varchar(10)                    collate latin1_general_bin
            , [Narration] Varchar(100)                    collate latin1_general_bin
            , [Reference] Varchar(30)                    collate latin1_general_bin
            );

--Placeholder to create indexes as required

        Create Table [#OriginalBatch]
            (
              [Lot] Varchar(50)                           collate
latin1_general_bin
            , [MasterJob] Varchar(20)                    collate latin1_general_bin
            , [DatabaseName] Varchar(120)                collate latin1_general_bin
            );
```

Author: Johnson, Chris

```
        Insert  [#OriginalBatch]
                ( [Lot]
                , [MasterJob]
                , [DatabaseName]
                )
                Select Distinct
                        [LT].[Lot]
                      , [MasterJob] = [LT].[JobPurchOrder]
                      , [LT].[DatabaseName]
                From    [#LotTransactionsDispatches] As [LT]
                Where   [LT].[TrnType] = 'R'
                        And [LT].[JobPurchOrder] <> '';
--script to combine base data and insert into results table

        Insert  [#ResultsDispatches]
                ( [Company]
                , [CompanyName]
                , [OriginalBatch]
                , [Lot]
                , [StockCode]
                , [Description]
                , [Customer]
                , [CustomerName]
                , [JobDescription]
                , [JobClassification]
                , [SellingPrice]
                , [SalesOrder]
                , [SalesOrderLine]
                , [TrnQuantity]
                , [TrnValue]
                , [TrnType]
                , [AmountModifier]
                , [TrnDate]
                , [OldExpiryDate]
                , [NewExpiryDate]
                , [Job]
                , [Bin]
                , [CustomerPoNumber]
                , [UnitCost]
                , [WarehouseDescription]
                , [StockUom]
                , [Narration]
                , [Reference]
                )
                Select  [Company] = [CN].[Company]
                      , [CompanyName] = [CN].[CompanyName]
                      , [OriginalBatch] = [OB].[MasterJob]
                      , [LT].[Lot]
                      , [LT].[StockCode]
                      , [IM].[Description]
                      , [LT].[Customer]
                      , [AC].[Name]
                      , [WM].[JobDescription]
                      , [WM].[JobClassification]
                      , [SellingPrice] = [SD].[MPrice]
                      , [LT].[SalesOrder]
                      , [LT].[SalesOrderLine]
```

Author: Johnson, Chris

```
                        , [LT].[TrnQuantity]
                        , [LT].[TrnValue]
                        , [LT].[TrnType]
                        , [TTAM].[AmountModifier]
                        , [TrnDate] = Convert(Date , [LT].[TrnDate])
                        , [LT].[OldExpiryDate]
                        , [LT].[NewExpiryDate]
                        , [LT].[Job]
                        , [LT].[Bin]
                        , [SM].[CustomerPoNumber]
                        , [LT].[UnitCost]
                        , [W].[WarehouseDescription]
                        , [IM].[StockUom]
                        , [LT].[Narration]
                        , [LT].[Reference]
                From    [#LotTransactionsDispatches] As [LT]
                        Left Join [#OriginalBatch] As [OB] On [OB].[Lot] =
[LT].[Lot]
                                                              And [OB].[Database-
Name] = [LT].[DatabaseName]
                        Left Join [#InvMasterDispatches] As [IM] On [IM].[StockCode]
= [LT].[StockCode]
                                                              And [IM].[Database-
Name] = [LT].[DatabaseName]
                        Left Join [#ArCustomerDispatches] As [AC] On [AC].[Customer]
= [LT].[Customer]
                                                              And [AC].[Database-
Name] = [LT].[DatabaseName]
                        Left Join [#WipMasterDispatches] As [WM] On [WM].[Job] =
[LT].[Job]
                                                              And [WM].[Database-
Name] = [LT].[DatabaseName]
                        Left Join [BlackBox].[Lookups].[TrnTypeAmountModifier]
                        As [TTAM] On [TTAM].[TrnType] = [LT].[TrnType]
                                     And [TTAM].[Company] = [LT].[DatabaseName]
                        Left Join [#SorMasterDispatches] As [SM] On [SM].[Sales-
Order] = [LT].[SalesOrder]
                                                              And [SM].[Database-
Name] = [LT].[DatabaseName]
                        Left Join [BlackBox].[Lookups].[Warehouse] As [W] On
[W].[Warehouse] = [LT].[Warehouse]
                                                              And [W].[Company] =
[LT].[DatabaseName]
                        Left Join [BlackBox].[Lookups].[CompanyNames] As [CN] On
[CN].[Company] = [LT].[DatabaseName]
                        Left Join [#SorDetailDispatches] As [SD] On [SD].[Sales-
Order] = [LT].[SalesOrder]
                                                              And [SD].[SalesOrder-
Line] = [LT].[SalesOrderLine]
                                                              And [SD].[Database-
Name] = [LT].[DatabaseName]
                Where   [LT].[TrnType] = 'D';

--return results

        Select  [R].[Company]
              , [R].[CompanyName]
              , [R].[OriginalBatch]
              , [R].[Lot]
              , [R].[StockCode]
              , [StockDescription] = [R].[Description]
```

Author: Johnson, Chris

```
                , [R].[Customer]
                , [R].[CustomerName]
                , [R].[JobDescription]
                , [R].[JobClassification]
                , [R].[SellingPrice]
                , [R].[SalesOrder]
                , [R].[SalesOrderLine]
                , [R].[TrnQuantity]
                , [R].[TrnValue]
                , [R].[TrnType]
                , [R].[AmountModifier]
                , [R].[TrnDate]
                , [R].[OldExpiryDate]
                , [R].[NewExpiryDate]
                , [R].[Job]
                , [R].[Bin]
                , [R].[CustomerPoNumber]
                , [R].[UnitCost]
                , [Warehouse] = [R].[WarehouseDescription]
                , [Uom] = [R].[StockUom]
                , [R].[Narration]
                , [R].[Reference]
                , [TranRank] = 99
                , [ContainerRank] = 99
        From    [#ResultsDispatches] As [R];

        Drop Table [#OriginalBatch];
        Drop Table [#ArCustomerDispatches];
        Drop Table [#InvMasterDispatches];
        Drop Table [#LotTransactionsDispatches];
        Drop Table [#SorMasterDispatches];
        Drop Table [#ResultsDispatches];
        Drop Table [#WipMasterDispatches];

    End;

GO
EXEC sp_addextendedproperty N'MS_Description', N'details of stock dispatched',
'SCHEMA', N'Report', 'PROCEDURE', N'UspResults_StockDispatches', NULL, NULL
GO
```

Uses

[Lookups].[CompanyNames]
[Lookups].[TrnTypeAmountModifier]
[Lookups].[Warehouse]
[Process].[ExecForEachDB]
[Process].[UspInsert_RedTagLogs]
[Report]

Used By

[Report].[UspResults_StockOutputDispatches]

Author: Johnson, Chris

[Report].[UspResults_StockLevels]

MS_Description

stock levels and locations

Parameters

Name	Data Type	Max Length (Bytes)
@Company	varchar(max)	max
@RedTagType	char	1
@RedTagUse	varchar(500)	500

SQL Script

```
CREATE Proc [Report].[UspResults_StockLevels]
    (
      @Company Varchar(Max)
    , @RedTagType Char(1)
    , @RedTagUse Varchar(500)
    )
As
    Begin
/*
Template designed by Chris Johnson, Prometic Group September 2015
Stored procedure set out to query multiple databases with the same information and
return it in a collated format
--exec [Report].[UspResults_StockLevels]  10
*/
        If IsNumeric(@Company) = 0
            Begin
                Select  @Company = Upper(@Company);
            End;

        Set NoCount On;

--Red tag

        Declare @RedTagDB Varchar(255)= Db_Name();
        Exec [Process].[UspInsert_RedTagLogs] @StoredProcDb = 'BlackBox' ,
            @StoredProcSchema = 'Report' ,
            @StoredProcName = 'UspResults_StockLevels' ,
            @UsedByType = @RedTagType , @UsedByName = @RedTagUse ,
            @UsedByDb = @RedTagDB;

--list the tables that are to be pulled back from each DB - if they are not found
the script will not be run against that db

        Declare @ListOfTables Varchar(Max) = 'InvWarehouse,InvMultBin,InvMaster,Inv-
FifoLifo';

--create temporary tables to be pulled from different databases, including a column
```

Author: Johnson, Chris

```
to id

        Create Table [#InvWarehouse]
            (
              [DatabaseName] Varchar(150) Collate Latin1_General_BIN
            , [Warehouse] Varchar(10) Collate Latin1_General_BIN
            , [StockCode] Varchar(35) Collate Latin1_General_BIN
            , [QtyOnHand] Numeric(20 , 7)
            , [UnitCost] Numeric(20 , 7)
            , [OpenBalCost1] Numeric(20 , 7)
            , [OpenBalCost2] Numeric(20 , 7)
            , [OpenBalCost3] Numeric(20 , 7)
            );
        Create Table [#InvMultBin]
            (
              [DatabaseName] Varchar(150) Collate Latin1_General_BIN
            , [Warehouse] Varchar(10) Collate Latin1_General_BIN
            , [QtyOnHand1] Numeric(20 , 7)
            , [QtyOnHand2] Numeric(20 , 7)
            , [QtyOnHand3] Numeric(20 , 7)
            , [StockCode] Varchar(35) Collate Latin1_General_BIN
            );
        Create Table [#InvMaster]
            (
              [DatabaseName] Varchar(150) Collate Latin1_General_BIN
            , [StockCode] Varchar(35) Collate Latin1_General_BIN
            , [Description] Varchar(150) Collate Latin1_General_BIN
            );
        Create Table [#InvFifoLifo]
            (
              [DatabaseName] Varchar(150) Collate Latin1_General_BIN
            , [Warehouse] Varchar(10) Collate Latin1_General_BIN
            , [StockCode] Varchar(35) Collate Latin1_General_BIN
            , [ReceiptQty] Numeric(20 , 7)
            , [UnitCost1] Numeric(20 , 7)
            , [UnitCost2] Numeric(20 , 7)
            , [UnitCost3] Numeric(20 , 7)
            , [QtyOnHand1] Numeric(20 , 7)
            , [QtyOnHand2] Numeric(20 , 7)
            , [QtyOnHand3] Numeric(20 , 7)
            );



--create script to pull data from each db into the tables

        Declare @SQL1 Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end'
            + --Only query DBs beginning SysProCompany

            '
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN'
            + --only companies selected in main run, or if companies selected then
```

```
all

            '
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT'
            + --count number of tables requested (number of commas plus one)

            '
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')'
            + --Count of the tables requested how many exist in the db

            '
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'','')) '
            + --only if the count matches (all the tables exist in the requested db)
then run the script

            '
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
                Insert [#InvWarehouse]
                        ( [DatabaseName]
                        , [Warehouse]
                        , [StockCode]
                        , [QtyOnHand]
                        , [UnitCost]
                        , [OpenBalCost1]
                        , [OpenBalCost2]
                        , [OpenBalCost3]
                        )
                SELECT [DatabaseName]=@DBCode
                     , [iw].[Warehouse]
                     , [iw].[StockCode]
                     , [iw].[QtyOnHand]
                     , [iw].[UnitCost]
                     , [iw].[OpenBalCost1]
                     , [iw].[OpenBalCost2]
                     , [iw].[OpenBalCost3]
                From [InvWarehouse] As [iw]
            End
    End';
        Declare @SQL2 Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end'
            + --Only query DBs beginning SysProCompany

            '
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN'
            + --only companies selected in main run, or if companies selected then
all
```

Author: Johnson, Chris

```
            '
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT'
            + --count number of tables requested (number of commas plus one)

            '
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')'
            + --Count of the tables requested how many exist in the db

            '
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'','')) '
            + --only if the count matches (all the tables exist in the requested db)
then run the script

            '
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
                Insert [#InvMultBin]
                        ( [DatabaseName]
                        , [Warehouse]
                        , [QtyOnHand1]
                        , [QtyOnHand2]
                        , [QtyOnHand3]
                        , [StockCode]
                        )
                SELECT [DatabaseName]=@DBCode
                     , [imb].[Warehouse]
                     , [imb].[QtyOnHand1]
                     , [imb].[QtyOnHand2]
                     , [imb].[QtyOnHand3]
                     , [imb].[StockCode]
                From [InvMultBin] As [imb]
            End
    End';
        Declare @SQL3 Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end'
            + --Only query DBs beginning SysProCompany

            '
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN'
            + --only companies selected in main run, or if companies selected then
all

            '
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables + '''
```

Author: Johnson, Chris

```
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT'
            + --count number of tables requested (number of commas plus one)

            '
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')'
            + --Count of the tables requested how many exist in the db

            '
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'','')) '
            + --only if the count matches (all the tables exist in the requested db)
then run the script

            '
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
                Insert [#InvMaster]
                        ( [DatabaseName]
                        , [StockCode]
                        , [Description]
                        )
                SELECT [DatabaseName]=@DBCode
                     , [im].[StockCode]
                     , [im].[Description]
                From [InvMaster] As [im]
            End
    End';
        Declare @SQL4 Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end'
            + --Only query DBs beginning SysProCompany

            '
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN'
            + --only companies selected in main run, or if companies selected then
all

            '
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT'
            + --count number of tables requested (number of commas plus one)

            '
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')'
            + --Count of the tables requested how many exist in the db

            '
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
```

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-StockLevels

```
Box.dbo.udf_SplitString(@ListOfTables,'','')) '
            + --only if the count matches (all the tables exist in the requested db)
then run the script

            '
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
            Insert [#InvFifoLifo]
                    ( [DatabaseName]
                    , [Warehouse]
                    , [StockCode]
                    , [ReceiptQty]
                    , [UnitCost1]
                    , [UnitCost2]
                    , [UnitCost3]
                    , [QtyOnHand1]
                    , [QtyOnHand2]
                    , [QtyOnHand3]
                    )
            SELECT [DatabaseName] = @DBCode
                 , [ifl].[Warehouse]
                 , [ifl].[StockCode]
                 , [ifl].[ReceiptQty]
                 , [ifl].[UnitCost1]
                 , [ifl].[UnitCost2]
                 , [ifl].[UnitCost3]
                 , [ifl].[QtyOnHand1]
                 , [ifl].[QtyOnHand2]
                 , [ifl].[QtyOnHand3]
            From [InvFifoLifo] As [ifl]
            End
    End';
--Enable this function to check script changes (try to run script directly against
db manually)

--Print @SQL


--execute script against each db, populating the base tables

        Exec [Process].[ExecForEachDB] @cmd = @SQL1;
        Exec [Process].[ExecForEachDB] @cmd = @SQL2;
        Exec [Process].[ExecForEachDB] @cmd = @SQL3;
        Exec [Process].[ExecForEachDB] @cmd = @SQL4;

--define the results you want to return


--Placeholder to create indexes as required


--script to combine base data and insert into results table


--return results

        Select  [cn].[CompanyName]
```

Author: Johnson, Chris

```
                , [ifl].[Warehouse]
                , [ifl].[StockCode]
                , [StockDescription] = [im].[Description]
                , [Receipted] = Sum([ifl].[ReceiptQty] * [ifl].[UnitCost1])
                , [Period1Value] = Sum([ifl].[UnitCost1] * [ifl].[QtyOnHand1])
                , [Period2Value] = Sum([ifl].[UnitCost2] * [ifl].[QtyOnHand2])
                , [Period3Value] = Sum([ifl].[UnitCost3] * [ifl].[QtyOnHand3])
                , [Period1Qty] = Sum([ifl].[QtyOnHand1])
                , [Period2Qty] = Sum([ifl].[QtyOnHand2])
                , [Period3Qty] = Sum([ifl].[QtyOnHand3])
                , [ifl].[UnitCost1]
                , [ifl].[UnitCost2]
                , [ifl].[UnitCost3]
        From    [#InvFifoLifo] As [ifl]
                Left Join [Lookups].[CompanyNames] As [cn]
                    On [ifl].[DatabaseName] = [cn].[Company]
                Left Join [#InvMaster] As [im]
                    On [im].[DatabaseName] = [ifl].[DatabaseName]
                       And [im].[StockCode] = [ifl].[StockCode]
        Group By [cn].[CompanyName]
                , [ifl].[Warehouse]
                , [ifl].[StockCode]
                , [im].[Description]
                , [ifl].[UnitCost1]
                , [ifl].[UnitCost2]
                , [ifl].[UnitCost3];

    End;

GO
EXEC sp_addextendedproperty N'MS_Description', N'stock levels and locations',
'SCHEMA', N'Report', 'PROCEDURE', N'UspResults_StockLevels', NULL, NULL
GO
```

Uses

[Lookups].[CompanyNames]
[Process].[ExecForEachDB]
[Process].[UspInsert_RedTagLogs]
[Report]

Author: Johnson, Chris

[Report].[UspResults_StockMovements]

MS_Description

list of all stock movements

Parameters

Name	Data Type	Max Length (Bytes)
@Company	varchar(max)	max
@RedTagType	char	1
@RedTagUse	varchar(500)	500

SQL Script

```
CREATE Proc [Report].[UspResults_StockMovements]
    (
      @Company Varchar(Max)
    , @RedTagType Char(1)
    , @RedTagUse Varchar(500)
    )
As
    Begin
/*
Template designed by Chris Johnson, Prometic Group September 2015
Stored procedure set out to query multiple databases with the same information and
return it in a collated format
-- exec [Report].[UspResults_StockMovements] 10 @Warehouses
='All',@Bins='All',@StockCodes='3000'
*/
        Set NoCount Off;
        If IsNumeric(@Company) = 0
            Begin
                Select  @Company = Upper(@Company);
            End;
        Declare @Warehouses Varchar(Max) = 'All'
          , @Bins Varchar(Max) = 'All'
          , @StockCodes Varchar(Max) = 'All';
--remove nocount on to speed up query

        Set NoCount On;

--Red tag

        Declare @RedTagDB Varchar(255)= Db_Name();
        Exec [Process].[UspInsert_RedTagLogs] @StoredProcDb = 'BlackBox' ,
            @StoredProcSchema = 'Report' ,
            @StoredProcName = 'UspResults_StockMovements' ,
            @UsedByType = @RedTagType , @UsedByName = @RedTagUse ,
            @UsedByDb = @RedTagDB;

--list the tables that are to be pulled back from each DB - if they are not found
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-StockMovements

```
the script will not be run against that db

        Declare @ListOfTables Varchar(Max) = 'InvMaster,InvMovements';

    --List of Warehouses selected by User

        Create Table [#WarehouseList]
            (
              [Warehouse] Varchar(10) Collate Latin1_General_BIN
            );

        If @Warehouses = 'All'
            Begin
                Insert  [#WarehouseList]
                        ( [Warehouse]
                        )
                        Select  [Warehouse]
                        From    [Lookups].[Warehouse]
                        Where   [Company] = @Company;
            End;
        If @Warehouses <> 'All'
            Begin
                Insert  [#WarehouseList]
                        ( [Warehouse]
                        )
                        Select  [Value]
                        From    [BlackBox].[dbo].[udf_SplitString](@Warehouses ,
                                                                   ',');
            End;

    --List of Bins selected by User

        Create Table [#BinList]
            (
              [Bin] Varchar(20) Collate Latin1_General_BIN
            );

        If @Bins = 'All'
            Begin
                Insert  [#BinList]
                        ( [Bin]
                        )
                        Select  [Bin]
                        From    [Lookups].[Bin]
                        Where   [Company] = @Company;
            End;
        If @Bins <> 'All'
            Begin
                Insert  [#BinList]
                        ( [Bin]
                        )
                        Select  [Value]
                        From    [BlackBox].[dbo].[udf_SplitString](@Bins , ',');
            End;

        Create --drop --alter
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-StockMovements

```
    Table [#StockCodeList]
            (
              [StockCode] Varchar(30) Collate Latin1_General_BIN
            );

        If @StockCodes = 'All'
            Begin
                Insert  [#StockCodeList]
                        ( [StockCode]
                        )
                        Select  [StockCode]
                        From    [Lookups].[StockCode]
                        Where   [Company] = @Company;
            End;
        If @StockCodes <> 'All'
            Begin
                Insert  [#StockCodeList]
                        ( [StockCode]
                        )
                        Select  [Value]
                        From    [BlackBox].[dbo].[udf_SplitString](@StockCodes ,
                                                                   ',');
            End;

--create temporary tables to be pulled from different databases, including a column
to id

        Create Table [#InvMaster]
            (
              [DatabaseName] Varchar(150)        collate latin1_general_bin
            , [AbcClass] Char(1)                 collate latin1_general_bin
            , [CostUom] Varchar(10)                collate latin1_general_bin
            , [CycleCount] Decimal
            , [Description] Varchar(50)            collate latin1_general_bin
            , [ProductClass] Varchar(20)         collate latin1_general_bin
            , [StockCode] Varchar(30)             collate latin1_general_bin
            , [StockUom] Varchar(10)             collate latin1_general_bin
            );



        Create Table [#InvMovements]
            (
              [DatabaseName] Varchar(150)        collate latin1_general_bin
            , [Bin] Varchar(20)                    collate latin1_general_bin
            , [EnteredCost] Float
            , [EntryDate] DateTime2
            , [MovementType] Char(1)             collate latin1_general_bin
            , [TrnQty] Float
            , [TrnType] Char(1)                    collate latin1_general_bin
            , [TrnValue] Float
            , [Warehouse] Varchar(10)             collate latin1_general_bin
            , [StockCode] Varchar(30)             collate latin1_general_bin
            , [LotSerial] Varchar(50)             collate latin1_general_bin
            , [TrnPeriod] Int
            );
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-StockMovements

```
--create script to pull data from each db into the tables

        Declare @SQL Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end'
            + --Only query DBs beginning SysProCompany

            '
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN'
            + --only companies selected in main run, or if companies selected then
all

            '
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT'
            + --count number of tables requested (number of commas plus one)

            '
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')'
            + --Count of the tables requested how many exist in the db

            '
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'','')) '
            + --only if the count matches (all the tables exist in the requested db)
then run the script

            '
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
            Insert #InvMaster
                ( DatabaseName
                , AbcClass
                , CostUom
                , CycleCount
                , Description
                , ProductClass
                , StockCode
                , StockUom
                )
            Select @DBCode
                , AbcClass
                , CostUom
                , CycleCount
                , Description
                , ProductClass
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-StockMovements

```
                , IM.StockCode
                , IM.StockUom
            From InvMaster IM
            inner join #StockCodeList SL
            on IM.StockCode = SL.StockCode

            Insert #InvMovements
                ( DatabaseName
                , Bin
                , EnteredCost
                , EntryDate
                , MovementType
                , TrnQty
                , TrnType
                , TrnValue
                , Warehouse
                , StockCode
                ,[LotSerial]
                , TrnPeriod
                )
            Select @DBCode
                ,    IM.Bin
                , EnteredCost
                , EntryDate
                , MovementType
                , TrnQty
                , TrnType
                , TrnValue
                , IM.Warehouse
                , IM.StockCode
                , IM.[LotSerial]
                , TrnPeriod = TrnYear*100+TrnMonth
            From dbo.InvMovements IM
            inner join #StockCodeList SL
                on IM.StockCode = SL.StockCode
            inner join #WarehouseList WL
                on IM.Warehouse = WL.Warehouse
            inner join #BinList BL
                on IM.Bin = BL.Bin
            End
    End';

--Enable this function to check script changes (try to run script directly against
db manually)

        Print @SQL;

--execute script against each db, populating the base tables

        Exec [Process].[ExecForEachDB] @cmd = @SQL;

--define the results you want to return

        Create Table [#Results]
            (
              [Company] Varchar(150)        collate latin1_general_bin
            , [AbcClass] Char(1)            collate latin1_general_bin
```

Author: Johnson, Chris

```
            , [CostUom] Varchar(10)            collate latin1_general_bin
            , [CycleCount] Float
            , [Description] Varchar(50)        collate latin1_general_bin
            , [ProductClass] Varchar(20)    collate latin1_general_bin
            , [StockCode] Varchar(30)        collate latin1_general_bin
            , [StockUom] Varchar(10)        collate latin1_general_bin
            , [Bin] Varchar(20)                collate latin1_general_bin
            , [EnteredCost] Decimal
            , [EntryDate] DateTime2
            , [MovementType] Char(1)        collate latin1_general_bin
            , [TrnQty] Float
            , [TrnType] Char(1)                collate latin1_general_bin
            , [TrnValue] Float
            , [Warehouse] Varchar(10)        collate latin1_general_bin
            , [LotSerial] Varchar(50)        collate latin1_general_bin
            , [TrnPeriod] Int
            );

--Placeholder to create indexes as required


--script to combine base data and insert into results table

        Insert  [#Results]
                ( [Company]
                , [AbcClass]
                , [CostUom]
                , [CycleCount]
                , [Description]
                , [ProductClass]
                , [StockCode]
                , [StockUom]
                , [Bin]
                , [EnteredCost]
                , [EntryDate]
                , [MovementType]
                , [TrnQty]
                , [TrnType]
                , [TrnValue]
                , [Warehouse]
                , [LotSerial]
                , [TrnPeriod]
                )
                Select  [Company] = [IM].[DatabaseName]
                      , [IM].[AbcClass]
                      , [IM].[CostUom]
                      , [IM].[CycleCount]
                      , [IM].[Description]
                      , [IM].[ProductClass]
                      , [IM].[StockCode]
                      , [IM].[StockUom]
                      , [IM2].[Bin]
                      , [IM2].[EnteredCost]
                      , [IM2].[EntryDate]
                      , [IM2].[MovementType]
                      , [IM2].[TrnQty]
                      , [IM2].[TrnType]
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-StockMovements

```
                      , [IM2].[TrnValue] * [TM].[AmountModifier]
                      , [IM2].[Warehouse]
                      , [IM2].[LotSerial]
                      , [IM2].[TrnPeriod]
                From    [#InvMaster] [IM]
                        Left Join [#InvMovements] [IM2] On [IM2].[DatabaseName] =
[IM].[DatabaseName]
                                                           And [IM2].[StockCode] =
[IM].[StockCode]
                        Left Join [Lookups].[TrnTypeAmountModifier] [TM] On
[TM].[TrnType] Collate Latin1_General_BIN = [IM2].[TrnType]
                                                             And [TM].[Company] =
[IM].[DatabaseName] Collate Latin1_General_BIN;

--return results

        Select  [Company]
              , [AbcClass]
              , [CostUom]
              , [CycleCount]
              , [Description]
              , [ProductClass]
              , [StockCode]
              , [StockUom]
              , [Bin]
              , [EnteredCost]
              , [EntryDate]
              , [MovementType]
              , [TrnQty]
              , [TrnType]
              , [TrnValue] = Coalesce([TrnValue] , 0)
              , [Warehouse]
              , [Lot] = Case When [LotSerial] = '' Then Null
                             When IsNumeric([LotSerial]) = 1
                             Then Cast([LotSerial] As BigInt)
                             Else [LotSerial]
                        End
              , [TrnPeriod] = Cast(DateAdd(Month ,
                                           Cast(Right([TrnPeriod] , 2) As Int)
                                           - 1 ,
                                           DateAdd(Year ,
                                                   Cast(Left([TrnPeriod] , 4) As
Int)
                                                   - 1900 ,
                                                   Cast('' As DateTime2))) As Date)
        From    [#Results];

    End;

GO
EXEC sp_addextendedproperty N'MS_Description', N'list of all stock movements',
'SCHEMA', N'Report', 'PROCEDURE', N'UspResults_StockMovements', NULL, NULL
GO
```

Uses

[Lookups].[Bin]

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-StockMovements

[Lookups].[StockCode]
[Lookups].[TrnTypeAmountModifier]
[Lookups].[Warehouse]
[Process].[ExecForEachDB]
[Process].[UspInsert_RedTagLogs]
[dbo].[udf_SplitString]
[Report]

Author: Johnson, Chris

[Report].[UspResults_StockOutput]

MS_Description

list of job output stock

Parameters

Name	Data Type	Max Length (Bytes)
@Company	varchar(max)	max
@RedTagType	char	1
@RedTagUse	varchar(500)	500

SQL Script

```
CREATE Proc [Report].[UspResults_StockOutput]
    (
      @Company Varchar(Max)
    , @RedTagType Char(1)
    , @RedTagUse Varchar(500)
    )
As
    Begin
/*
Template designed by Chris Johnson, Prometic Group September 2015
Stored procedure set out to query multiple databases with the same information and
return it in a collated format
Details of where lots are distributed
--exec [Report].[UspResults_StockOutput]  10
*/
        If IsNumeric(@Company) = 0
            Begin
                Select  @Company = Upper(@Company);
            End;


--remove nocount on to speed up query

        Set NoCount On;
--Red tag

        Declare @RedTagDB Varchar(255)= Db_Name();
        Exec [Process].[UspInsert_RedTagLogs] @StoredProcDb = 'BlackBox' ,
            @StoredProcSchema = 'Report' ,
            @StoredProcName = 'UspResults_StockOutput' ,
            @UsedByType = @RedTagType , @UsedByName = @RedTagUse ,
            @UsedByDb = @RedTagDB;
--list the tables that are to be pulled back from each DB - if they are not found
the script will not be run against that db

        Declare @ListOfTables Varchar(Max) = 'LotTransactions,WipMaster,SorMaster';
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-StockOutput

```
--create temporary tables to be pulled from different databases, including a column
to id

        Create Table [#LotTransactions]
            (
              [DatabaseName] Varchar(150)    Collate Latin1_General_BIN
            , [JobPurchOrder] Varchar(50)    Collate Latin1_General_BIN
            , [Lot] Varchar(50)                Collate Latin1_General_BIN
            , [TrnQuantity] Numeric(20 , 7)
            , [TrnValue] Numeric(20 , 2)
            , [TrnType] Varchar(5)            Collate Latin1_General_BIN
            , [TrnDate] DateTime2
            , [OldExpiryDate] DateTime2
            , [NewExpiryDate] DateTime2
            , [Job] Varchar(50)                Collate Latin1_General_BIN
            , [Bin] Varchar(50)                Collate Latin1_General_BIN
            , [UnitCost] Numeric(20 , 7)
            , [Warehouse] Varchar(10)        Collate Latin1_General_BIN
            , [Narration] Varchar(150)        Collate Latin1_General_BIN
            , [Reference] Varchar(150)        Collate Latin1_General_BIN
            );
        Create Table [#WipMaster]
            (
              [DatabaseName] Varchar(150)    Collate Latin1_General_BIN
            , [Customer] Varchar(50)        Collate Latin1_General_BIN
            , [CustomerName] Varchar(255)    Collate Latin1_General_BIN
            , [Job] Varchar(50)                Collate Latin1_General_BIN
            , [JobDescription] Varchar(255) Collate Latin1_General_BIN
            , [JobClassification] Varchar(50) Collate Latin1_General_BIN
            , [SellingPrice] Numeric(20 , 2)
            , [SalesOrder] Varchar(50)        Collate Latin1_General_BIN
            , [SalesOrderLine] Varchar(15) Collate Latin1_General_BIN
            );
        Create Table [#Lots]
            (
              [DatabaseName] Varchar(150)        collate latin1_general_bin
            , [JobPurchOrder] Varchar(50)        collate latin1_general_bin
            , [Lot] Varchar(50)                    collate latin1_general_bin
            , [StockCode] Varchar(50)            collate latin1_general_bin
            , [StockDescription] Varchar(255)    collate latin1_general_bin
            );
        Create Table [#SorMaster]
            (
              [DatabaseName] Varchar(150)        Collate Latin1_General_BIN
            , [SalesOrder] Varchar(50)            Collate Latin1_General_BIN
            , [CustomerPoNumber] Varchar(150)    collate latin1_general_bin
            );
        Create Table [#InvMaster]
            (
              [DatabaseName] Varchar(150)        collate latin1_general_bin
            , [StockCode] Varchar(50)            collate latin1_general_bin
            , [StockUom] Varchar(10)            collate latin1_general_bin
            );
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-StockOutput

```
--create script to pull data from each db into the tables

        Declare @SQL1 Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end'
            + --Only query DBs beginning SysProCompany

            '
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN'
            + --only companies selected in main run, or if companies selected then
all

            '
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT'
            + --count number of tables requested (number of commas plus one)

            '
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')'
            + --Count of the tables requested how many exist in the db

            '
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'','')) '
            + --only if the count matches (all the tables exist in the requested db)
then run the script

            '
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
            Insert [#LotTransactions]
                    ( [DatabaseName], [JobPurchOrder]
                    , [Lot], [TrnQuantity]
                    , [TrnValue], [TrnType]
                    , [TrnDate], [OldExpiryDate], [NewExpiryDate]
                    , [Job], [Bin]
                    , [UnitCost], [Warehouse]
                    , [Narration], [Reference]
                    )
            SELECT [DatabaseName] = @DBCode
                    , [lt].[JobPurchOrder]
                    , [lt].[Lot], [lt].[TrnQuantity]
                    , [lt].[TrnValue], [lt].[TrnType]
                    , [lt].[TrnDate], [lt].[OldExpiryDate], [lt].[NewExpiryDate]
                    , [lt].[Job], [lt].[Bin]
                    , [lt].[UnitCost], [Warehouse]
                    , [Narration], [Reference]
            FROM [LotTransactions] As [lt]
            Insert [#Lots]
                    ( [DatabaseName]
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-StockOutput

```
                    , [JobPurchOrder], [Lot]
                    , StockCode, StockDescription
                    )
            Select
                    Distinct
                [DatabaseName] = @DBCode
              , [l].[JobPurchOrder], [l].[Lot]
              , [l].[StockCode], im.[Description]
            From
                [LotTransactions] As [l]
            Left Join [dbo].[InvMaster] As [im]
                On [im].[StockCode] = [l].[StockCode]
            Where
                [JobPurchOrder] <> ''''
            End
    End';
        Declare @SQL2 Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end'
            + --Only query DBs beginning SysProCompany

            '
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN'
            + --only companies selected in main run, or if companies selected then
all

            '
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT'
            + --count number of tables requested (number of commas plus one)

            '
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')'
            + --Count of the tables requested how many exist in the db

            '
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'','')) '
            + --only if the count matches (all the tables exist in the requested db)
then run the script

            '
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
                Insert [#WipMaster]
                        ( [DatabaseName]
                        , [Customer]
                        , [CustomerName]
                        , [Job]
                        , [JobDescription]
```

Author: Johnson, Chris

```
                        , [JobClassification]
                        , [SellingPrice]
                        , [SalesOrder]
                        , [SalesOrderLine]
                        )
                SELECT [DatabaseName]=@DBCode
                     , [wm].[Customer]
                     , [wm].[CustomerName]
                     , [wm].[Job]
                     , [wm].[JobDescription]
                     , [wm].[JobClassification]
                     , [wm].[SellingPrice]
                     , [SalesOrder]
                     , [SalesOrderLine]
                FROM [WipMaster] As [wm]
            End
    End';
        Declare @SQL3 Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end'
            + --Only query DBs beginning SysProCompany

            '
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN'
            + --only companies selected in main run, or if companies selected then
all

            '
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT'
            + --count number of tables requested (number of commas plus one)

            '
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')'
            + --Count of the tables requested how many exist in the db

            '
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'','')) '
            + --only if the count matches (all the tables exist in the requested db)
then run the script

            '
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
            Insert [#SorMaster]
                ( [DatabaseName]
                , [SalesOrder]
                , [CustomerPoNumber]
                )
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-StockOutput

```
            SELECT [DatabaseName]=@DBCode
                 , [sm].[SalesOrder]
                 , [sm].[CustomerPoNumber]
            FROM [SorMaster] As [sm]
            End
    End';
        Declare @SQL4 Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end'
            + --Only query DBs beginning SysProCompany

            '
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN'
            + --only companies selected in main run, or if companies selected then
all

            '
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT'
            + --count number of tables requested (number of commas plus one)

            '
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')'
            + --Count of the tables requested how many exist in the db

            '
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'','')) '
            + --only if the count matches (all the tables exist in the requested db)
then run the script

            '
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
            Insert [#InvMaster]
                ( [DatabaseName]
                , [StockCode]
                , [StockUom]
                )
            SELECT [DatabaseName]=@DBCode
                 , [im].[StockCode]
                 , [im].[StockUom]
            From [InvMaster] As [im]
            End
    End';

--Enable this function to check script changes (try to run script directly against
db manually)

--Print @SQL
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-StockOutput

```
--execute script against each db, populating the base tables

        Exec [Process].[ExecForEachDB] @cmd = @SQL1;
        Exec [Process].[ExecForEachDB] @cmd = @SQL2;
        Exec [Process].[ExecForEachDB] @cmd = @SQL3;
        Exec [Process].[ExecForEachDB] @cmd = @SQL4;
--define the results you want to return

        Create Table [#Results]
            (
              [DatabaseName] Varchar(150)            collate latin1_general_bin
            , [CompanyName] Varchar(150)            collate latin1_general_bin
            , [JobPurchOrder] Varchar(50)           collate latin1_general_bin
            , [Lot] Varchar(50)                       collate latin1_general_bin
            , [StockCode] Varchar(50)               collate latin1_general_bin
            , [StockDescription] Varchar(255)       collate latin1_general_bin
            , [Customer] Varchar(50)                collate latin1_general_bin
            , [CustomerName] Varchar(255)           collate latin1_general_bin
            , [JobDescription] Varchar(255)           collate latin1_general_bin
            , [JobClassification] Varchar(150)      collate latin1_general_bin
            , [SellingPrice] Numeric(20 , 2)
            , [SalesOrder] Varchar(50)                collate latin1_general_bin
            , [SalesOrderLine] Varchar(15)            collate latin1_general_bin
            , [TrnQuantity] Numeric(20 , 7)
            , [TrnValue] Numeric(20 , 2)
            , [TrnType] Varchar(100)                collate latin1_general_bin
            , [AmountModifier] Int
            , [TrnDate] DateTime2
            , [OldExpiryDate] DateTime2
            , [NewExpiryDate] DateTime2
            , [Job] Varchar(50)                       collate latin1_general_bin
            , [Bin] Varchar(50)                       collate latin1_general_bin
            , [CustomerPoNumber] Varchar(150)       collate latin1_general_bin
            , [UnitCost] Numeric(20 , 7)
            , [StockUom] Varchar(10)                collate latin1_general_bin
            , [Warehouse] Varchar(200)                collate latin1_general_bin
            , [Narration] Varchar(150)                Collate Latin1_General_BIN
            , [Reference] Varchar(150)                Collate Latin1_General_BIN
            );

--Placeholder to create indexes as required

--create NonClustered Index Index_Name On #Table1 (DatabaseName) Include (Column-Name)


--script to combine base data and insert into results table

        Insert  [#Results]
                ( [DatabaseName]
                , [CompanyName]
                , [JobPurchOrder]
                , [Lot]
                , [StockCode]
                , [StockDescription]
```

Author: Johnson, Chris

```
                , [Customer]
                , [CustomerName]
                , [JobDescription]
                , [JobClassification]
                , [SellingPrice]
                , [SalesOrder]
                , [SalesOrderLine]
                , [TrnQuantity]
                , [TrnValue]
                , [TrnType]
                , [AmountModifier]
                , [TrnDate]
                , [OldExpiryDate]
                , [NewExpiryDate]
                , [Job]
                , [Bin]
                , [CustomerPoNumber]
                , [UnitCost]
                , [StockUom]
                , [Warehouse]
                , [Narration]
                , [Reference]
                )
                Select  [lt].[DatabaseName]
                      , [cn].[CompanyName]
                      , [l].[JobPurchOrder]
                      , [l].[Lot]
                      , [l].[StockCode]
                      , [l].[StockDescription]
                      , [wm].[Customer]
                      , [wm].[CustomerName]
                      , [wm].[JobDescription]
                      , [wm].[JobClassification]
                      , [wm].[SellingPrice]
                      , [wm].[SalesOrder]
                      , [wm].[SalesOrderLine]
                      , [lt].[TrnQuantity]
                      , [lt].[TrnValue]
                      , [TrnType] = [LTTT].[TrnTypeDescription]--[lt].[TrnType]

                      , [ttam].[AmountModifier]
                      , [lt].[TrnDate]
                      , [lt].[OldExpiryDate]
                      , [lt].[NewExpiryDate]
                      , [lt].[Job]
                      , [lt].[Bin]
                      , [sm].[CustomerPoNumber]
                      , [lt].[UnitCost]
                      , [im].[StockUom]
                      , [w].[WarehouseDescription]
                      , [lt].[Narration]
                      , [lt].[Reference]
                From    [#LotTransactions] As [lt]
                        Inner Join [#Lots] As [l]
                            On [l].[Lot] = [lt].[Lot] Collate Latin1_General_BIN
                               And [l].[DatabaseName] = [lt].[DatabaseName] Collate
Latin1_General_BIN
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-StockOutput

```
                        Left Join [#WipMaster] As [wm]
                            On [wm].[Job] = [lt].[Job] Collate Latin1_General_BIN
                               And [wm].[DatabaseName] = [lt].[DatabaseName] Collate
Latin1_General_BIN
                        Left Join [#SorMaster] As [sm]
                            On [sm].[SalesOrder] = [wm].[SalesOrder]
                               And [sm].[DatabaseName] = [wm].[DatabaseName]
                        Left Join [#InvMaster] As [im]
                            On [im].[DatabaseName] = [lt].[DatabaseName]
                               And [im].[StockCode] = [l].[StockCode]
                        Left Join [BlackBox].[Lookups].[TrnTypeAmountModifier]
                            As [ttam]
                            On [ttam].[TrnType] = [lt].[TrnType]  Collate Latin1_-
General_BIN
                               And [ttam].[Company] = [lt].[DatabaseName] Collate
Latin1_General_BIN
                        Left Join [Lookups].[CompanyNames] As [cn]
                            On [cn].[Company] = [lt].[DatabaseName] Collate Latin1_-
General_BIN
                        Left Join [BlackBox].[Lookups].[Warehouse] As [w]
                            On [w].[Warehouse] = [lt].[Warehouse]
                               And [w].[Company] = [lt].[DatabaseName]
                        Left Join [Lookups].[LotTransactionTrnType] [LTTT]
                            On [LTTT].[TrnType] = [lt].[TrnType];

--return results

        Select  [Company] = [DatabaseName]
              , [CompanyName]
              , [OriginalBatch] = [JobPurchOrder]
              , [Lot]
              , [StockCode]
              , [StockDescription]
              , [Customer]
              , [CustomerName]
              , [JobDescription]
              , [JobClassification]
              , [SellingPrice]
              , [SalesOrder]
              , [SalesOrderLine]
              , [TrnQuantity]
              , [TrnValue]
              , [TrnType]
              , [AmountModifier] = Coalesce([AmountModifier] , 0)
              , [TrnDate] = Cast([TrnDate] As Date)
              , [OldExpiryDate] = Cast([OldExpiryDate] As Date)
              , [NewExpiryDate] = Cast([NewExpiryDate] As Date)
              , [Job]
              , [Bin]
              , [CustomerPoNumber]
              , [UnitCost]
              , [Warehouse]
              , [Uom] = [StockUom]
              , [Narration] = Case When [Narration] = '' Then Null
                                   Else [Narration]
                              End
              , [Reference] = Case When [Reference] = '' Then Null
                                   Else [Reference]
```

Author: Johnson, Chris

```
                                End
                , [TranRank] = Rank() Over ( Partition By [Lot] Order By [TrnDate] Asc
)
                , [ContainerRank] = Dense_Rank() Over ( Partition By [JobPurchOrder]
Order By [Lot] Asc )
        From    [#Results];


    End;


GO
EXEC sp_addextendedproperty N'MS_Description', N'list of job output stock',
'SCHEMA', N'Report', 'PROCEDURE', N'UspResults_StockOutput', NULL, NULL
GO
```

Uses

[Lookups].[CompanyNames]
[Lookups].[LotTransactionTrnType]
[Lookups].[TrnTypeAmountModifier]
[Lookups].[Warehouse]
[Process].[ExecForEachDB]
[Process].[UspInsert_RedTagLogs]
[Report]

Used By

[Report].[UspResults_StockOutputDispatches]

Author: Johnson, Chris

[Report].[UspResults_StockOutputDispatches]

MS_Description

not in use

Parameters

Name	Data Type	Max Length (Bytes)
@Company	varchar(max)	max
@RedTagType	char	1
@RedTagUse	varchar(500)	500

SQL Script

```
CREATE Proc [Report].[UspResults_StockOutputDispatches]
    (
      @Company Varchar(Max)
    , @RedTagType Char(1)
    , @RedTagUse Varchar(500)
    )
As
    Begin
/*
Template designed by Chris Johnson, Prometic Group March 2016

*/
        If IsNumeric(@Company) = 0
            Begin
                Select  @Company = Upper(@Company);
            End;

        --Red tag

        Declare @RedTagDB Varchar(255)= Db_Name();
        Exec [BlackBox].[Process].[UspInsert_RedTagLogs] @StoredProcDb = 'BlackBox'
,
            @StoredProcSchema = 'Report' ,
            @StoredProcName = 'UspResults_StockOutputDispatches' ,
            @UsedByType = @RedTagType , @UsedByName = @RedTagUse ,
            @UsedByDb = @RedTagDB;

        Declare @Company2 Varchar(Max)
          , @RedTagType2 Char(1)
          , @RedTagUse2 Varchar(500);

        Set @Company2 = @Company;
        Set @RedTagType2 = @RedTagType;
        Set @RedTagUse2 = @RedTagUse;

        Create Table [#Results]
            (
```

Author: Johnson, Chris

```
            [Company] Varchar(150)                collate latin1_general_bin
          , [CompanyName] Varchar(150)            collate latin1_general_bin
          , [OriginalBatch] Varchar(50)            collate latin1_general_bin
          , [Lot] Varchar(50)                        collate latin1_general_bin
          , [StockCode] Varchar(50)                collate latin1_general_bin
          , [StockDescription] Varchar(255)        collate latin1_general_bin
          , [Customer] Varchar(50)                 collate latin1_general_bin
          , [CustomerName] Varchar(255)            collate latin1_general_bin
          , [JobDescription] Varchar(255)            collate latin1_general_bin
          , [JobClassification] Varchar(150)       collate latin1_general_bin
          , [SellingPrice] Numeric(20 , 2)
          , [SalesOrder] Varchar(50)               collate latin1_general_bin
          , [SalesOrderLine] Varchar(15)           collate latin1_general_bin
          , [TrnQuantity] Numeric(20 , 8)
          , [TrnValue] Numeric(20 , 2)
          , [TrnType] Varchar(100)
          , [AmountModifier] Int
          , [TrnDate] Date
          , [OldExpiryDate] Date
          , [NewExpiryDate] Date
          , [Job] Varchar(150)                     collate latin1_general_bin
          , [Bin] Varchar(150)                     collate latin1_general_bin
          , [CustomerPoNumber] Varchar(150)        collate latin1_general_bin
          , [UnitCost] Numeric(20 , 7)
          , [Warehouse] Varchar(200)               collate latin1_general_bin
          , [Uom] Varchar(10)                        collate latin1_general_bin
          , [Narration] Varchar(500)               collate latin1_general_bin
          , [Reference] Varchar(500)               collate latin1_general_bin
          , [TranRank] BigInt
          , [ContainerRank] BigInt
          );

      Insert  [#Results]
              ( [Company]
              , [CompanyName]
              , [OriginalBatch]
              , [Lot]
              , [StockCode]
              , [StockDescription]
              , [Customer]
              , [CustomerName]
              , [JobDescription]
              , [JobClassification]
              , [SellingPrice]
              , [SalesOrder]
              , [SalesOrderLine]
              , [TrnQuantity]
              , [TrnValue]
              , [TrnType]
              , [AmountModifier]
              , [TrnDate]
              , [OldExpiryDate]
              , [NewExpiryDate]
              , [Job]
              , [Bin]
              , [CustomerPoNumber]
              , [UnitCost]
```

Author: Johnson, Chris

```
                , [Warehouse]
                , [Uom]
                , [Narration]
                , [Reference]
                , [TranRank]
                , [ContainerRank]
                )
                Exec [BlackBox].[Report].[UspResults_StockOutput] @Company =
@Company2 ,
                    @RedTagType = @RedTagType2 , @RedTagUse = @RedTagUse2;

        Insert  [#Results]
                ( [Company]
                , [CompanyName]
                , [OriginalBatch]
                , [Lot]
                , [StockCode]
                , [StockDescription]
                , [Customer]
                , [CustomerName]
                , [JobDescription]
                , [JobClassification]
                , [SellingPrice]
                , [SalesOrder]
                , [SalesOrderLine]
                , [TrnQuantity]
                , [TrnValue]
                , [TrnType]
                , [AmountModifier]
                , [TrnDate]
                , [OldExpiryDate]
                , [NewExpiryDate]
                , [Job]
                , [Bin]
                , [CustomerPoNumber]
                , [UnitCost]
                , [Warehouse]
                , [Uom]
                , [Narration]
                , [Reference]
                , [TranRank]
                , [ContainerRank]
                )
                Exec [BlackBox].[Report].[UspResults_StockDispatches] @Company =
@Company2 ,
                    @RedTagType = @RedTagType2 , @RedTagUse = @RedTagUse2;

        Select  [R].[Company]
              , [R].[CompanyName]
              , [R].[OriginalBatch]
              , [R].[Lot]
              , [R].[StockCode]
              , [R].[StockDescription]
              , [R].[Customer]
              , [R].[CustomerName]
              , [R].[JobDescription]
              , [R].[JobClassification]
```

Author: Johnson, Chris

```
                , [R].[SellingPrice]
                , [R].[SalesOrder]
                , [R].[SalesOrderLine]
                , [R].[TrnQuantity]
                , [R].[TrnValue]
                , [R].[TrnType]
                , [R].[AmountModifier]
                , [R].[TrnDate]
                , [R].[OldExpiryDate]
                , [R].[NewExpiryDate]
                , [R].[Job]
                , [R].[Bin]
                , [R].[CustomerPoNumber]
                , [R].[UnitCost]
                , [R].[Warehouse]
                , [R].[Uom]
                , [R].[Narration]
                , [R].[Reference]
                , [R].[TranRank]
                , [R].[ContainerRank]
        From    [#Results] As [R];

        Drop Table [#Results];
    End;
GO
EXEC sp_addextendedproperty N'MS_Description', N'not in use', 'SCHEMA', N'Report',
'PROCEDURE', N'UspResults_StockOutputDispatches', NULL, NULL
GO
```

Uses

[Process].[UspInsert_RedTagLogs]
[Report].[UspResults_StockDispatches]
[Report].[UspResults_StockOutput]
[Report]

Author: Johnson, Chris

[Report].[UspResults_StockOutputDispatches2]

MS_Description

not in use

Parameters

Name	Data Type	Max Length (Bytes)
@Company	varchar(50)	50
@RedTagType	char	1
@RedTagUse	varchar(500)	500

SQL Script

```
CREATE Proc [Report].[UspResults_StockOutputDispatches2]
    (
      @Company Varchar(50)
    , @RedTagType Char(1)
    , @RedTagUse Varchar(500)
    )
As
    Begin
/*
Template designed by Chris Johnson, Prometic Group March 2016

*/
        If IsNumeric(@Company) = 0
            Begin
                Select  @Company = Upper(@Company);
            End;

        --Red tag

        Declare @RedTagDB Varchar(255)= Db_Name();
        Exec [BlackBox].[Process].[UspInsert_RedTagLogs] @StoredProcDb = 'BlackBox'
,
            @StoredProcSchema = 'Report' ,
            @StoredProcName = 'UspResults_StockOutputDispatches2' ,
            @UsedByType = @RedTagType , @UsedByName = @RedTagUse ,
            @UsedByDb = @RedTagDB;

    --Dispatches

    --define the results you want to return

        Create Table [#ResultsDispatches]
            (
              [Company] Varchar(150)                        collate
latin1_general_bin
            , [CompanyName] Varchar(250)                    collate
latin1_general_bin
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-StockOutputDispatches2

```
            , [OriginalBatch] Varchar(20)                    collate
latin1_general_bin
            , [Lot] Varchar(50)                               collate
latin1_general_bin
            , [StockCode] Varchar(30)                        collate
latin1_general_bin
            , [Description] Varchar(50)                       collate
latin1_general_bin
            , [Customer] Varchar(15)                         collate
latin1_general_bin
            , [CustomerName] Varchar(50)                     collate
latin1_general_bin
            , [JobDescription] Varchar(50)                    collate
latin1_general_bin
            , [JobClassification] Varchar(10)                collate
latin1_general_bin
            , [SellingPrice] Numeric(20 , 2)
            , [SalesOrder] Varchar(20)                        collate
latin1_general_bin
            , [SalesOrderLine] Int
            , [TrnQuantity] Numeric(20 , 8)
            , [TrnValue] Numeric(20 , 2)
            , [TrnType] Char(1)                               collate
latin1_general_bin
            , [AmountModifier] Float
            , [TrnDate] Date
            , [OldExpiryDate] Date
            , [NewExpiryDate] Date
            , [Job] Varchar(20)                               collate
latin1_general_bin
            , [Bin] Varchar(20)                               collate
latin1_general_bin
            , [CustomerPoNumber] Varchar(30)                 collate
latin1_general_bin
            , [UnitCost] Numeric(20 , 2)
            , [WarehouseDescription] Varchar(200)            collate
latin1_general_bin
            , [StockUom] Varchar(10)                         collate
latin1_general_bin
            , [Narration] Varchar(100)                        collate
latin1_general_bin
            , [Reference] Varchar(30)                        collate
latin1_general_bin
            );
            --define the results you want to return

        Create Table [#Results]
            (
              [DatabaseName] Varchar(150)                    collate
latin1_general_bin
            , [CompanyName] Varchar(150)                     collate
latin1_general_bin
            , [JobPurchOrder] Varchar(50)                    collate
latin1_general_bin
            , [Lot] Varchar(50)                               collate
latin1_general_bin
            , [StockCode] Varchar(50)                        collate
latin1_general_bin
            , [StockDescription] Varchar(255)                collate
latin1_general_bin
            , [Customer] Varchar(50)                         collate
latin1_general_bin
            , [CustomerName] Varchar(255)                    collate
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-StockOutputDispatches2

```
latin1_general_bin
            , [JobDescription] Varchar(255)                    collate
latin1_general_bin
            , [JobClassification] Varchar(150)                 collate
latin1_general_bin
            , [SellingPrice] Numeric(20 , 2)
            , [SalesOrder] Varchar(50)                         collate
latin1_general_bin
            , [SalesOrderLine] Varchar(15)                     collate
latin1_general_bin
            , [TrnQuantity] Numeric(20 , 7)
            , [TrnValue] Numeric(20 , 2)
            , [TrnType] Varchar(10)                            collate
latin1_general_bin
            , [AmountModifier] Int
            , [TrnDate] DateTime2
            , [OldExpiryDate] DateTime2
            , [NewExpiryDate] DateTime2
            , [Job] Varchar(50)                                collate
latin1_general_bin
            , [Bin] Varchar(50)                                collate
latin1_general_bin
            , [CustomerPoNumber] Varchar(150)                 collate
latin1_general_bin
            , [UnitCost] Numeric(20 , 7)
            , [StockUom] Varchar(10)                          collate
latin1_general_bin
            , [Warehouse] Varchar(200)                         collate
latin1_general_bin
            , [Narration] Varchar(150)                         collate
latin1_general_bin
            , [Reference] Varchar(150)                         collate
latin1_general_bin
            );
        Begin
        --list the tables that are to be pulled back from each DB - if they are not
found the script will not be run against that db

            Declare @ListOfTables Varchar(Max) = 'AssetDepreciation,TblApTerms';

--create temporary tables to be pulled from different databases, including a column
to id

            Create Table [#LotTransactionsDispatches]
                (
                  [DatabaseName] Varchar(150)                 collate
latin1_general_bin
                , [Lot] Varchar(50)                            collate
latin1_general_bin
                , [StockCode] Varchar(30)                     collate
latin1_general_bin
                , [Customer] Varchar(15)                      collate
latin1_general_bin
                , [SalesOrder] Varchar(20)                     collate
latin1_general_bin
                , [SalesOrderLine] Int
                , [TrnQuantity] Numeric(20 , 8)
                , [TrnValue] Numeric(20 , 2)
                , [TrnType] Char(1)                            collate
latin1_general_bin
                , [TrnDate] Date
                , [OldExpiryDate] Date
```

Author: Johnson, Chris

```
                , [NewExpiryDate] Date
                , [Job] Varchar(20)                                collate
latin1_general_bin
                , [Bin] Varchar(20)                                collate
latin1_general_bin
                , [UnitCost] Numeric(20 , 2)
                , [Narration] Varchar(100)                        collate
latin1_general_bin
                , [Reference] Varchar(30)                        collate
latin1_general_bin
                , [Warehouse] Varchar(10)                        collate
latin1_general_bin
                , [JobPurchOrder] Varchar(20)                    collate
latin1_general_bin
                );
            Create Table [#InvMasterDispatches]
                (
                  [DatabaseName] Varchar(150)                    collate
latin1_general_bin
                , [StockCode] Varchar(30)                        collate
latin1_general_bin
                , [Description] Varchar(50)                        collate
latin1_general_bin
                , [StockUom] Varchar(10)                        collate
latin1_general_bin
                );
            Create Table [#ArCustomerDispatches]
                (
                  [DatabaseName] Varchar(150)                    collate
latin1_general_bin
                , [Customer] Varchar(15)                        collate
latin1_general_bin
                , [Name] Varchar(50)                            collate
latin1_general_bin
                );
            Create Table [#WipMasterDispatches]
                (
                  [DatabaseName] Varchar(150)                    collate
latin1_general_bin
                , [Job] Varchar(20)                                collate
latin1_general_bin
                , [JobDescription] Varchar(50)                    collate
latin1_general_bin
                , [JobClassification] Varchar(10)                collate
latin1_general_bin
                , [SellingPrice] Numeric(20 , 2)
                );
            Create Table [#SorMasterDispatches]
                (
                  [DatabaseName] Varchar(150)                    collate
latin1_general_bin
                , [SalesOrder] Varchar(20)                        collate
latin1_general_bin
                , [CustomerPoNumber] Varchar(30)                collate
latin1_general_bin
                );
            Create Table [#SorDetailDispatches]
                (
                  [DatabaseName] Varchar(150)                    collate
latin1_general_bin
                , [SalesOrder] Varchar(20)                        collate
latin1_general_bin
```

Author: Johnson, Chris

```
                , [SalesOrderLine] Int
                , [MPrice] Numeric(20 , 2)
                );

--create script to pull data from each db into the tables

            Declare @SQLLotTransactions Varchar(Max) = 'USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
                + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables
                + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'',''))
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
                Insert [#LotTransactionsDispatches]
                        ( [DatabaseName]
                        , [Lot]
                        , [StockCode]
                        , [Customer]
                        , [SalesOrder]
                        , [SalesOrderLine]
                        , [TrnQuantity]
                        , [TrnValue]
                        , [TrnType]
                        , [TrnDate]
                        , [OldExpiryDate]
                        , [NewExpiryDate]
                        , [Job]
                        , [Bin]
                        , [UnitCost]
                        , [Narration]
                        , [Reference]
                        , [Warehouse]
                        , [JobPurchOrder]
                        )
                SELECT [DatabaseName]=@DBCode
                     , [LT].[Lot]
                     , [LT].[StockCode]
                     , [LT].[Customer]
                     , [LT].[SalesOrder]
                     , [LT].[SalesOrderLine]
                     , [LT].[TrnQuantity]
                     , [LT].[TrnValue]
                     , [LT].[TrnType]
                     , [LT].[TrnDate]
                     , [LT].[OldExpiryDate]
                     , [LT].[NewExpiryDate]
```

Author: Johnson, Chris

```
                    , [LT].[Job]
                    , [LT].[Bin]
                    , [LT].[UnitCost]
                    , [LT].[Narration]
                    , [LT].[Reference]
                    , [LT].[Warehouse]
                    , [LT].[JobPurchOrder] FROM [LotTransactions] As [LT]
            End
    End';
            Declare @SQLInvMaster Varchar(Max) = 'USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
                + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables
                + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'',''))
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
                Insert [#InvMasterDispatches]
                        ( [DatabaseName]
                        , [StockCode]
                        , [Description]
                        , [StockUom]
                        )
                SELECT [DatabaseName]=@DBCode
                     , [IM].[StockCode]
                     , [IM].[Description]
                     , [IM].[StockUom] FROM [InvMaster] As [IM]
            End
    End';
            Declare @SQLArCustomer Varchar(Max) = 'USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
                + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables
                + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'',''))
            If @ActualCountOfTables=@RequiredCountOfTables
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-StockOutputDispatches2

```
            BEGIN
            Insert [#ArCustomerDispatches]
                    ( [DatabaseName]
                    , [Customer]
                    , [Name]
                    )
            SELECT [DatabaseName]=@DBCode
                 , [AC].[Customer]
                 , [AC].[Name] FROM [ArCustomer] As [AC]
            End
    End';
            Declare @SQLWipMaster Varchar(Max) = 'USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
                + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables
                + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'',''))
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
            Insert [#WipMasterDispatches]
                    ( [DatabaseName]
                    , [Job]
                    , [JobDescription]
                    , [JobClassification]
                    , [SellingPrice]
                    )
            SELECT [DatabaseName]=@DBCode
                 , [WM].[Job]
                 , [WM].[JobDescription]
                 , [WM].[JobClassification]
                 , [WM].[SellingPrice] FROM [WipMaster] As [WM]
            End
    End';
            Declare @SQLSorMaster Varchar(Max) = 'USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
                + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables
                + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-StockOutputDispatches2

```
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'',''))
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
            Insert [#SorMasterDispatches]
                    ( [DatabaseName]
                    , [SalesOrder]
                    , [CustomerPoNumber]
                    )
            SELECT [DatabaseName]=@DBCode
                 , [SM].[SalesOrder]
                 , [SM].[CustomerPoNumber] FROM [SorMaster] As [SM]
            End
    End';
            Declare @SQLSorDetail Varchar(Max) = 'USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables
                + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'',''))
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
            Insert [#SorDetailDispatches]
                ( [DatabaseName]
                , [SalesOrder]
                , [SalesOrderLine]
                , [MPrice]
                )
            Select  [DatabaseName]=@DBCode
                  , [SD].[SalesOrder]
                  , [SD].[SalesOrderLine]
                  , [SD].[MPrice]
            From    [SorDetail] As [SD]
            End
    End';
--Enable this function to check script changes (try to run script directly against
db manually)

--Print @SQL


--execute script against each db, populating the base tables

            Begin
                Exec [BlackBox].[Process].[ExecForEachDB] @cmd = @SQLLot-
Transactions;
```

Author: Johnson, Chris

```
                Exec [BlackBox].[Process].[ExecForEachDB] @cmd = @SQLInvMaster;
                Exec [BlackBox].[Process].[ExecForEachDB] @cmd = @SQLArCustomer;
                Exec [BlackBox].[Process].[ExecForEachDB] @cmd = @SQLWipMaster;
                Exec [BlackBox].[Process].[ExecForEachDB] @cmd = @SQLSorMaster;
                Exec [BlackBox].[Process].[ExecForEachDB] @cmd = @SQLSorDetail;
            End;

--Placeholder to create indexes as required

            Create Table [#OriginalBatch]
                (
                  [Lot] Varchar(50)                collate latin1_general_bin
                , [MasterJob] Varchar(20)        collate latin1_general_bin
                , [DatabaseName] Varchar(120)    collate latin1_general_bin
                );
            Insert  [#OriginalBatch]
                    ( [Lot]
                    , [MasterJob]
                    , [DatabaseName]
                    )
                    Select Distinct
                            [LT].[Lot]
                          , [MasterJob] = [LT].[JobPurchOrder]
                          , [LT].[DatabaseName]
                    From    [#LotTransactionsDispatches] As [LT]
                    Where   [LT].[TrnType] = 'R'
                            And [LT].[JobPurchOrder] <> '';
--script to combine base data and insert into results table

            Insert  [#ResultsDispatches]
                    ( [Company]
                    , [CompanyName]
                    , [OriginalBatch]
                    , [Lot]
                    , [StockCode]
                    , [Description]
                    , [Customer]
                    , [CustomerName]
                    , [JobDescription]
                    , [JobClassification]
                    , [SellingPrice]
                    , [SalesOrder]
                    , [SalesOrderLine]
                    , [TrnQuantity]
                    , [TrnValue]
                    , [TrnType]
                    , [AmountModifier]
                    , [TrnDate]
                    , [OldExpiryDate]
                    , [NewExpiryDate]
                    , [Job]
                    , [Bin]
                    , [CustomerPoNumber]
                    , [UnitCost]
                    , [WarehouseDescription]
                    , [StockUom]
                    , [Narration]
```

Author: Johnson, Chris

```
                    , [Reference]
                    )
                    Select  [Company] = [CN].[Company]
                          , [CompanyName] = [CN].[CompanyName]
                          , [OriginalBatch] = [OB].[MasterJob]
                          , [LT].[Lot]
                          , [LT].[StockCode]
                          , [IM].[Description]
                          , [LT].[Customer]
                          , [AC].[Name]
                          , [WM].[JobDescription]
                          , [WM].[JobClassification]
                          , [SellingPrice] = [SD].[MPrice]
                          , [LT].[SalesOrder]
                          , [LT].[SalesOrderLine]
                          , [LT].[TrnQuantity]
                          , [LT].[TrnValue]
                          , [LT].[TrnType]
                          , [TTAM].[AmountModifier]
                          , [TrnDate] = Convert(Date , [LT].[TrnDate])
                          , [LT].[OldExpiryDate]
                          , [LT].[NewExpiryDate]
                          , [LT].[Job]
                          , [LT].[Bin]
                          , [SM].[CustomerPoNumber]
                          , [LT].[UnitCost]
                          , [W].[WarehouseDescription]
                          , [IM].[StockUom]
                          , [LT].[Narration]
                          , [LT].[Reference]
                    From    [#LotTransactionsDispatches] As [LT]
                            Left Join [#OriginalBatch] As [OB] On [OB].[Lot] =
[LT].[Lot]
                                                                  And [OB].[Database-
Name] = [LT].[DatabaseName]
                            Left Join [#InvMasterDispatches] As [IM] On [IM].[Stock-
Code] = [LT].[StockCode]
                                                                  And [IM].[Database-
Name] = [LT].[DatabaseName]
                            Left Join [#ArCustomerDispatches] As [AC] On
[AC].[Customer] = [LT].[Customer]
                                                                  And [AC].[Database-
Name] = [LT].[DatabaseName]
                            Left Join [#WipMasterDispatches] As [WM] On [WM].[Job] =
[LT].[Job]
                                                                  And [WM].[Database-
Name] = [LT].[DatabaseName]
                            Left Join [BlackBox].[Lookups].[TrnTypeAmountModifier]
                            As [TTAM] On [TTAM].[TrnType] = [LT].[TrnType]
                                         And [TTAM].[Company] = [LT].[DatabaseName]
                            Left Join [#SorMasterDispatches] As [SM] On [SM].[Sales-
Order] = [LT].[SalesOrder]
                                                                  And [SM].[Database-
Name] = [LT].[DatabaseName]
                            Left Join [BlackBox].[Lookups].[Warehouse] As [W] On
[W].[Warehouse] = [LT].[Warehouse]
                                                                  And [W].[Company] =
[LT].[DatabaseName]
                            Left Join [BlackBox].[Lookups].[CompanyNames] As [CN] On
[CN].[Company] = [LT].[DatabaseName]
```

Author: Johnson, Chris

```
                            Left Join [#SorDetailDispatches] As [SD] On [SD].[Sales-
Order] = [LT].[SalesOrder]
                                                                    And [SD].[SalesOrder-
Line] = [LT].[SalesOrderLine]
                                                                    And [SD].[Database-
Name] = [LT].[DatabaseName]
                    Where   [LT].[TrnType] = 'D';



            Drop Table [#OriginalBatch];
            Drop Table [#ArCustomerDispatches];
            Drop Table [#InvMasterDispatches];
            Drop Table [#LotTransactionsDispatches];
            Drop Table [#SorMasterDispatches];
            Drop Table [#WipMasterDispatches];
        End;


    --Output

        Begin
        --create temporary tables to be pulled from different databases, including a
column to id

            Create Table [#LotTransactions]
                (
                  [DatabaseName] Varchar(150)        Collate Latin1_General_BIN
                , [JobPurchOrder] Varchar(50)        Collate Latin1_General_BIN
                , [Lot] Varchar(50)                    Collate Latin1_General_BIN
                , [TrnQuantity] Numeric(20 , 7)
                , [TrnValue] Numeric(20 , 2)
                , [TrnType] Varchar(5)                Collate Latin1_General_BIN
                , [TrnDate] DateTime2
                , [OldExpiryDate] DateTime2
                , [NewExpiryDate] DateTime2
                , [Job] Varchar(50)                    Collate Latin1_General_BIN
                , [Bin] Varchar(50)                    Collate Latin1_General_BIN
                , [UnitCost] Numeric(20 , 7)
                , [Warehouse] Varchar(10)            Collate Latin1_General_BIN
                , [Narration] Varchar(150)            Collate Latin1_General_BIN
                , [Reference] Varchar(150)            Collate Latin1_General_BIN
                );
            Create Table [#WipMaster]
                (
                  [DatabaseName] Varchar(150)        Collate Latin1_General_BIN
                , [Customer] Varchar(50)            Collate Latin1_General_BIN
                , [CustomerName] Varchar(255)        Collate Latin1_General_BIN
                , [Job] Varchar(50)                    Collate Latin1_General_BIN
                , [JobDescription] Varchar(255)        Collate Latin1_General_BIN
                , [JobClassification] Varchar(50)    Collate Latin1_General_BIN
                , [SellingPrice] Numeric(20 , 2)
                , [SalesOrder] Varchar(50)            Collate Latin1_General_BIN
                , [SalesOrderLine] Varchar(15)        Collate Latin1_General_BIN
                );
            Create Table [#Lots]
                (
                  [DatabaseName] Varchar(150)        collate latin1_general_bin
```

```
                , [JobPurchOrder] Varchar(50)         collate latin1_general_bin
                , [Lot] Varchar(50)                    collate latin1_general_bin
                , [StockCode] Varchar(50)             collate latin1_general_bin
                , [StockDescription] Varchar(255)     collate latin1_general_bin
                );
            Create Table [#SorMaster]
                (
                  [DatabaseName] Varchar(150)         Collate Latin1_General_BIN
                , [SalesOrder] Varchar(50)             Collate Latin1_General_BIN
                , [CustomerPoNumber] Varchar(150)     collate latin1_general_bin
                );
            Create Table [#InvMaster]
                (
                  [DatabaseName] Varchar(150)         collate latin1_general_bin
                , [StockCode] Varchar(50)             collate latin1_general_bin
                , [StockUom] Varchar(10)             collate latin1_general_bin
                );

--create script to pull data from each db into the tables

            Declare @SQL1 Varchar(Max) = 'USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
              + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables + '''
                  , @RequiredCountOfTables INT
                  , @ActualCountOfTables INT
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'',''))
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
            Insert [#LotTransactions]
                  ( [DatabaseName], [JobPurchOrder]
                  , [Lot], [TrnQuantity]
                  , [TrnValue], [TrnType]
                  , [TrnDate], [OldExpiryDate], [NewExpiryDate]
                  , [Job], [Bin]
                  , [UnitCost], [Warehouse]
                  , [Narration], [Reference]
                  )
            SELECT [DatabaseName] = @DBCode
                  , [lt].[JobPurchOrder]
                  , [lt].[Lot], [lt].[TrnQuantity]
                  , [lt].[TrnValue], [lt].[TrnType]
                  , [lt].[TrnDate], [lt].[OldExpiryDate], [lt].[NewExpiryDate]
                  , [lt].[Job], [lt].[Bin]
                  , [lt].[UnitCost], [Warehouse]
                  , [Narration], [Reference]
            FROM [LotTransactions] As [lt]
            Insert [#Lots]
                  ( [DatabaseName]
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-StockOutputDispatches2

```
                    , [JobPurchOrder], [Lot]
                    , StockCode, StockDescription
                    )
            Select
                    Distinct
                [DatabaseName] = @DBCode
              , [l].[JobPurchOrder], [l].[Lot]
              , [l].[StockCode], im.[Description]
            From
                [LotTransactions] As [l]
            Left Join [dbo].[InvMaster] As [im]
                On [im].[StockCode] = [l].[StockCode]
            Where
                [JobPurchOrder] <> ''''
            End
    End';
            Declare @SQL2 Varchar(Max) = 'USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
                + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'',''))
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
                Insert [#WipMaster]
                        ( [DatabaseName]
                        , [Customer]
                        , [CustomerName]
                        , [Job]
                        , [JobDescription]
                        , [JobClassification]
                        , [SellingPrice]
                        , [SalesOrder]
                        , [SalesOrderLine]
                        )
                SELECT [DatabaseName]=@DBCode
                     , [wm].[Customer]
                     , [wm].[CustomerName]
                     , [wm].[Job]
                     , [wm].[JobDescription]
                     , [wm].[JobClassification]
                     , [wm].[SellingPrice]
                     , [SalesOrder]
                     , [SalesOrderLine]
                FROM [WipMaster] As [wm]
            End
    End';
            Declare @SQL3 Varchar(Max) = 'USE [?];
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-StockOutputDispatches2

```
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
                + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'',''))
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
            Insert [#SorMaster]
                ( [DatabaseName]
                , [SalesOrder]
                , [CustomerPoNumber]
                )
            SELECT [DatabaseName]=@DBCode
                 , [sm].[SalesOrder]
                 , [sm].[CustomerPoNumber]
            FROM [SorMaster] As [sm]
            End
    End';
            Declare @SQL4 Varchar(Max) = 'USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
                + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'',''))
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
            Insert [#InvMaster]
                ( [DatabaseName]
                , [StockCode]
                , [StockUom]
                )
            SELECT [DatabaseName]=@DBCode
                 , [im].[StockCode]
                 , [im].[StockUom]
            From [InvMaster] As [im]
            End
    End';

--Enable this function to check script changes (try to run script directly against
```

Author: Johnson, Chris

```
db manually)

--Print @SQL


--execute script against each db, populating the base tables

            Exec [BlackBox].[Process].[ExecForEachDB] @cmd = @SQL1;
            Exec [BlackBox].[Process].[ExecForEachDB] @cmd = @SQL2;
            Exec [BlackBox].[Process].[ExecForEachDB] @cmd = @SQL3;
            Exec [BlackBox].[Process].[ExecForEachDB] @cmd = @SQL4;


--Placeholder to create indexes as required

--create NonClustered Index Index_Name On #Table1 (DatabaseName) Include (Column-
Name)


--script to combine base data and insert into results table

            Insert  [#Results]
                    ( [DatabaseName]
                    , [CompanyName]
                    , [JobPurchOrder]
                    , [Lot]
                    , [StockCode]
                    , [StockDescription]
                    , [Customer]
                    , [CustomerName]
                    , [JobDescription]
                    , [JobClassification]
                    , [SellingPrice]
                    , [SalesOrder]
                    , [SalesOrderLine]
                    , [TrnQuantity]
                    , [TrnValue]
                    , [TrnType]
                    , [AmountModifier]
                    , [TrnDate]
                    , [OldExpiryDate]
                    , [NewExpiryDate]
                    , [Job]
                    , [Bin]
                    , [CustomerPoNumber]
                    , [UnitCost]
                    , [StockUom]
                    , [Warehouse]
                    , [Narration]
                    , [Reference]
                    )
                    Select  [lt].[DatabaseName]
                          , [cn].[CompanyName]
                          , [l].[JobPurchOrder]
                          , [l].[Lot]
                          , [l].[StockCode]
                          , [l].[StockDescription]
```

Author: Johnson, Chris

```
                          , [wm].[Customer]
                          , [wm].[CustomerName]
                          , [wm].[JobDescription]
                          , [wm].[JobClassification]
                          , [wm].[SellingPrice]
                          , [wm].[SalesOrder]
                          , [wm].[SalesOrderLine]
                          , [lt].[TrnQuantity]
                          , [lt].[TrnValue]
                          , [lt].[TrnType]
                          , [ttam].[AmountModifier]
                          , [lt].[TrnDate]
                          , [lt].[OldExpiryDate]
                          , [lt].[NewExpiryDate]
                          , [lt].[Job]
                          , [lt].[Bin]
                          , [sm].[CustomerPoNumber]
                          , [lt].[UnitCost]
                          , [im].[StockUom]
                          , [w].[WarehouseDescription]
                          , [lt].[Narration]
                          , [lt].[Reference]
                  From    [#LotTransactions] As [lt]
                          Inner Join [#Lots] As [l] On [l].[Lot] = [lt].[Lot]
Collate Latin1_General_BIN
                                                     And [l].[DatabaseName] =
[lt].[DatabaseName] Collate Latin1_General_BIN
                          Left Join [#WipMaster] As [wm] On [wm].[Job] =
[lt].[Job] Collate Latin1_General_BIN
                                                     And [wm].[Database-
Name] = [lt].[DatabaseName] Collate Latin1_General_BIN
                          Left Join [#SorMaster] As [sm] On [sm].[SalesOrder] =
[wm].[SalesOrder]
                                                     And [sm].[Database-
Name] = [wm].[DatabaseName]
                          Left Join [#InvMaster] As [im] On [im].[DatabaseName] =
[lt].[DatabaseName]
                                                     And [im].[StockCode] =
[l].[StockCode]
                          Left Join [BlackBox].[Lookups].[TrnTypeAmountModifier]
                          As [ttam] On [ttam].[TrnType] = [lt].[TrnType]  Collate
Latin1_General_BIN
                                               And [ttam].[Company] = [lt].[DatabaseName]
Collate Latin1_General_BIN
                          Left Join [BlackBox].[Lookups].[CompanyNames] As [cn] On
[cn].[Company] = [lt].[DatabaseName] Collate Latin1_General_BIN
                          Left Join [BlackBox].[Lookups].[Warehouse] As [w] On
[w].[Warehouse] = [lt].[Warehouse]
                                                     And [w].[Company] =
[lt].[DatabaseName];


        End;
        --return results

        Select  [R].[Company]
              , [R].[CompanyName]
              , [R].[OriginalBatch]
              , [R].[Lot]
              , [R].[StockCode]
```

Author: Johnson, Chris

```
              , [StockDescription] = [R].[Description]
              , [R].[Customer]
              , [R].[CustomerName]
              , [R].[JobDescription]
              , [R].[JobClassification]
              , [R].[SellingPrice]
              , [R].[SalesOrder]
              , [R].[SalesOrderLine]
              , [R].[TrnQuantity]
              , [R].[TrnValue]
              , [R].[TrnType]
              , [R].[AmountModifier]
              , [R].[TrnDate]
              , [R].[OldExpiryDate]
              , [R].[NewExpiryDate]
              , [R].[Job]
              , [R].[Bin]
              , [R].[CustomerPoNumber]
              , [R].[UnitCost]
              , [Warehouse] = [R].[WarehouseDescription]
              , [Uom] = [R].[StockUom]
              , [R].[Narration]
              , [R].[Reference]
              , [TranRank] = 99
              , [ContainerRank] = 99
       From   [#ResultsDispatches] As [R]
       Union all
       Select [Company] = [DatabaseName]
              , [CompanyName]
              , [OriginalBatch] = [JobPurchOrder]
              , [Lot]
              , [StockCode]
              , [StockDescription]
              , [Customer]
              , [CustomerName]
              , [JobDescription]
              , [JobClassification]
              , [SellingPrice]
              , [SalesOrder]
              , [SalesOrderLine]
              , [TrnQuantity]
              , [TrnValue]
              , [TrnType]
              , [AmountModifier] = Coalesce([AmountModifier] , 0)
              , [TrnDate] = Cast([TrnDate] As Date)
              , [OldExpiryDate] = Cast([OldExpiryDate] As Date)
              , [NewExpiryDate] = Cast([NewExpiryDate] As Date)
              , [Job]
              , [Bin]
              , [CustomerPoNumber]
              , [UnitCost]
              , [Warehouse]
              , [Uom] = [StockUom]
              , [Narration] = Case When [Narration] = '' Then Null
                                   Else [Narration]
                              End
              , [Reference] = Case When [Reference] = '' Then Null
```

Author: Johnson, Chris

```
                                    Else [Reference]
                                End
              , [TranRank] = Rank() Over ( Partition By [Lot] Order By [TrnDate] Asc
)
              , [ContainerRank] = Dense_Rank() Over ( Partition By [JobPurchOrder]
Order By [Lot] Asc )
        From    [#Results];
    End;
GO
EXEC sp_addextendedproperty N'MS_Description', N'not in use', 'SCHEMA', N'Report',
'PROCEDURE', N'UspResults_StockOutputDispatches2', NULL, NULL
GO
```

Uses

[Lookups].[CompanyNames]
[Lookups].[TrnTypeAmountModifier]
[Lookups].[Warehouse]
[Process].[ExecForEachDB]
[Process].[UspInsert_RedTagLogs]
[Report]

[Report].[UspResults_StockWithRunTimes]

MS_Description

list of stock, including the labour time to generate stock

Parameters

Name	Data Type	Max Length (Bytes)
@Company	varchar(max)	max
@EndPeriod	date	3
@LotLowerNumber	bigint	8
@RedTagType	char	1
@RedTagUse	varchar(500)	500

SQL Script

```
CREATE Proc [Report].[UspResults_StockWithRunTimes]
    (
      @Company Varchar(Max)
    , @EndPeriod Date
    , @LotLowerNumber BigInt
    , @RedTagType Char(1)
    , @RedTagUse Varchar(500)
    )
As
    Begin

        Set NoCount On;

        If IsNumeric(@Company) = 0
            Begin
                Select  @Company = Upper(@Company);
            End;

--Red tag

        Declare @RedTagDB Varchar(255)= Db_Name();
        Exec [Process].[UspInsert_RedTagLogs] @StoredProcDb = 'BlackBox' ,
            @StoredProcSchema = 'Report' ,
            @StoredProcName = 'UspResults_Template' ,
            @UsedByType = @RedTagType , @UsedByName = @RedTagUse ,
            @UsedByDb = @RedTagDB;



--list the tables that are to be pulled back from each DB - if they are not found
the script will not be run against that db

        Declare @ListOfTables Varchar(Max) = 'WipLabJnl,LotTransactions,InvMaster';
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-StockWithRunTimes

```
--create temporary tables to be pulled from different databases, including a column
to id



        Create Table [#WipLabJnl]
            (
              [DatabaseCode] Varchar(10)    collate latin1_general_bin
            , [Job] Varchar(20)               collate latin1_general_bin
            , [RunTime] Numeric(20 , 4)
            );
        Create Table [#LotTransactions]
            (
              [DatabaseCode] Varchar(10)    collate latin1_general_bin
            , [StockCode] Varchar(30)        collate latin1_general_bin
            , [Warehouse] Varchar(10)        collate latin1_general_bin
            , [TrnQuantity] Numeric(20 , 6)
            , [JobPurchOrder] Varchar(20)    collate latin1_general_bin
            , [TrnDate] Date
            , [TrnType] Char(1) Collate Latin1_General_BIN
            , [Lot] Varchar(50)               collate latin1_general_bin
            );
        Create Table [#InvMaster]
            (
              [DatabaseCode] Varchar(10)    collate latin1_general_bin
            , [StockCode] Varchar(30)        collate latin1_general_bin
            , [Description] Varchar(50)       collate latin1_general_bin
            );
--temp tables used in calculations

        Create Table [#WarehouseLevels]
            (
              [DatabaseCode] Varchar(10)    collate latin1_general_bin
            , [StockCode] Varchar(30)        collate latin1_general_bin
            , [Warehouse] Varchar(10)        collate latin1_general_bin
            , [TrnQuantityMod] Numeric(20 , 8)
            , [JobPurchOrder] Varchar(30)    collate latin1_general_bin
            , [CreatedDate] Date
            );
        Create Table [#HoursPerJob]
            (
              [DatabaseCode] Varchar(10)    collate latin1_general_bin
            , [Job] Varchar(20)               collate latin1_general_bin
            , [RunTime] Numeric(20 , 4)
            );

--create script to pull data from each db into the tables

        Declare @SQLInvMaster Varchar(Max) = 'USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            BEGIN
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-StockWithRunTimes

```
                Insert  [#InvMaster]
                ( [DatabaseCode]
                , [StockCode]
                , [Description]
                )
                Select  [DatabaseCode] = @DBCode
                      , [IM].[StockCode]
                      , [IM].[Description]
                From    [InvMaster] [IM];
            End
    End';
        Declare @SQLLotTransactions Varchar(Max) = 'USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            BEGIN
                Insert  [#LotTransactions]
                ( [DatabaseCode]
                , [StockCode]
                , [Warehouse]
                , [TrnQuantity]
                , [JobPurchOrder]
                , [TrnDate]
                , [TrnType]
                , [Lot]
                )
                Select  [DatabaseCode] = @DBCode
                      , [LT].[StockCode]
                      , [LT].[Warehouse]
                      , [LT].[TrnQuantity]
                      , [LT].[JobPurchOrder]
                      , [LT].[TrnDate]
                      , [LT].[TrnType]
                      , [LT].[Lot]
                From    [LotTransactions] [LT];
            End
    End';
        Declare @SQLWipLabJnl Varchar(Max) = 'USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            BEGIN
                Insert  [#WipLabJnl]
                ( [DatabaseCode]
                , [Job]
                , [RunTime]
                )
                Select  [DatabaseCode] = @DBCode
                      , [WLJ].[Job]
```

Author: Johnson, Chris

```
                    , [WLJ].[RunTime]
                From    [WipLabJnl] [WLJ];
            End
    End';

--execute script against each db, populating the base tables

        Exec [Process].[ExecForEachDB_WithTableCheck] @cmd = @SQLInvMaster ,
            @SchemaTablesToCheck = @ListOfTables;
        Exec [Process].[ExecForEachDB_WithTableCheck] @cmd = @SQLLotTransactions ,
            @SchemaTablesToCheck = @ListOfTables;
        Exec [Process].[ExecForEachDB_WithTableCheck] @cmd = @SQLWipLabJnl ,
            @SchemaTablesToCheck = @ListOfTables;

--define the results you want to return

        Create Table [#Results]
            (
              [DatabaseCode] Varchar(10)        collate latin1_general_bin
            , [StockCode] Varchar(30)            collate latin1_general_bin
            , [StockDescription] Varchar(150)    collate latin1_general_bin
            , [Warehouse] Varchar(10)            collate latin1_general_bin
            , [QtyOnHand] Numeric(20 , 8)
            , [RunTimeTotal] Numeric(20 , 2)
            , [TimePerUnit] Float
            , [JobQuantity] Numeric(20 , 8)
            , [JobPurchOrder] Varchar(30)        collate latin1_general_bin
            );

--Placeholder to create indexes as required

        --Get time logged per job

        Insert  [#HoursPerJob]
                ( [DatabaseCode]
                , [Job]
                , [RunTime]
                )
                Select  [WLJ].[DatabaseCode]
                      , [WLJ].[Job]
                      , Sum([WLJ].[RunTime])
                From    [#WipLabJnl] [WLJ]
                Group By [WLJ].[DatabaseCode]
                      , [WLJ].[Job];

        --get amount per job in each warehouse

        Insert  [#WarehouseLevels]
                ( [DatabaseCode]
                , [StockCode]
                , [Warehouse]
                , [TrnQuantityMod]
                , [JobPurchOrder]
                , [CreatedDate]
                )
                Select  [LT].[DatabaseCode]
                      , [LT].[StockCode]
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-StockWithRunTimes

```
                      , [LT].[Warehouse]
                      , [TrnQuantityMod] = Sum([LT].[TrnQuantity]
                                               * [TTAM].[AmountModifier])
                      , [LT2].[JobPurchOrder]
                      , [CreatedDate] = [LT2].[TrnDate]
                From    [#LotTransactions] [LT]
                        Inner Join [BlackBox].[Lookups].[TrnTypeAmountModifier]
[TTAM]
                            On [TTAM].[TrnType] = [LT].[TrnType]
                               And [TTAM].[Company] = [LT].[DatabaseCode]
                        Left Join [#LotTransactions] [LT2]
                            On [LT].[Lot] = [LT2].[Lot]
                               And [LT2].[DatabaseCode] = [LT].[DatabaseCode]
                               And [LT2].[TrnType] = 'R'
                Where   [LT].[TrnDate] <= @EndPeriod
                        And Case When IsNumeric([LT].[Lot]) = 1
                                 Then Convert(Numeric(20) , [LT].[Lot])
                                 Else @LotLowerNumber + 1
                            End > @LotLowerNumber
                Group By [LT].[DatabaseCode]
                      , [LT].[StockCode]
                      , [LT].[Warehouse]
                      , [LT2].[JobPurchOrder]
                      , [LT2].[TrnDate];

        Select  [LT].[DatabaseCode]
              , [LT].[JobPurchOrder]
              , [TimePerUnit] = Case When [HPJ].[RunTime] Is Null
                                     Then Convert(Float , 0)
                                     When [HPJ].[RunTime] = 0
                                     Then Convert(Float , 0)
                                     Else Convert(Float , [HPJ].[RunTime])
                                          / Convert(Float , Sum([LT].[TrnQuantity]))
                                End
              , [JobQuantity] = Sum([LT].[TrnQuantity])
        Into    [#Test]
        From    [#LotTransactions] [LT]
                Left Join [#HoursPerJob] [HPJ]
                    On [HPJ].[DatabaseCode] = [LT].[DatabaseCode]
                       And [LT].[JobPurchOrder] = [HPJ].[Job]
        Where   [LT].[TrnType] = 'R'
        Group By [LT].[DatabaseCode]
              , [LT].[JobPurchOrder]
              , [HPJ].[RunTime];

--script to combine base data and insert into results table

        Insert  [#Results]
                ( [DatabaseCode]
                , [StockCode]
                , [StockDescription]
                , [Warehouse]
                , [QtyOnHand]
                , [RunTimeTotal]
                , [TimePerUnit]
                , [JobQuantity]
                , [JobPurchOrder]
```

Author: Johnson, Chris

```
                )
                Select  [WL].[DatabaseCode]
                      , [WL].[StockCode]
                      , [StockDescription] = [IM].[Description]
                      , [WL].[Warehouse]
                      , [QtyOnHand] = Sum([WL].[TrnQuantityMod])
                      , [RunTimeTotal] = Sum([HPJ].[RunTime])
                      , [T].[TimePerUnit]
                      , [JobQuantity] = Sum([T].[JobQuantity])
                      , [WL].[JobPurchOrder]
                From    [#WarehouseLevels] [WL]
                        Left Join [#HoursPerJob] [HPJ]
                            On [HPJ].[Job] = [WL].[JobPurchOrder]
                        Left Join [#InvMaster] [IM]
                            On [IM].[StockCode] = [WL].[StockCode]
                               And [IM].[DatabaseCode] = [WL].[DatabaseCode]
                        Left Join [#Test] [T]
                            On [T].[DatabaseCode] = [WL].[DatabaseCode]
                               And [T].[JobPurchOrder] = [WL].[JobPurchOrder]
                Group By [WL].[DatabaseCode]
                      , [WL].[StockCode]
                      , [IM].[Description]
                      , [WL].[Warehouse]
                      , [T].[TimePerUnit]
                      , [WL].[JobPurchOrder]
                      --, [T].[JobQuantity]

                Having  Sum([WL].[TrnQuantityMod]) <> 0;

        Drop Table [#HoursPerJob];
        Drop Table [#InvMaster];
        Drop Table [#LotTransactions];
        Drop Table [#WarehouseLevels];
        Drop Table [#WipLabJnl];

        Set NoCount Off;
--return results

        Select  [R].[DatabaseCode]
              , [R].[StockCode]
              , [R].[StockDescription]
              , [R].[Warehouse]
              , [R].[QtyOnHand]
              , [R].[RunTimeTotal]
              , [R].[TimePerUnit]
              , [R].[JobQuantity]
              , [CN].[CompanyName]
              , [CN].[ShortName]
              , [MasterJob] = Case When IsNumeric([R].[JobPurchOrder]) = 1
                                   Then Convert(Varchar(30) , Convert(BigInt ,
[R].[JobPurchOrder]))
                                   Else [R].[JobPurchOrder]
                              End
        From    [#Results] [R]
                Left Join [Lookups].[CompanyNames] [CN]
                    On [R].[DatabaseCode] = [CN].[Company];
```

Author: Johnson, Chris

```
    End;


GO
EXEC sp_addextendedproperty N'MS_Description', N'list of stock, including the labour time to generate stock', 'SCHEMA', N'Report', 'PROCEDURE', N'UspResults_StockWithRun-Times', NULL, NULL
GO
```

Uses

[Lookups].[CompanyNames]
[Lookups].[TrnTypeAmountModifier]
[Process].[ExecForEachDB_WithTableCheck]
[Process].[UspInsert_RedTagLogs]
[Report]

[Report].[UspResults_Template]

MS_Description

used to develop new stored procs (copy and paste code and alter as necessary)

Parameters

Name	Data Type	Max Length (Bytes)
@Company	varchar(max)	max
@RedTagType	char	1
@RedTagUse	varchar(500)	500

SQL Script

```
CREATE Proc [Report].[UspResults_Template]
 ( @Company Varchar(Max)
    , @RedTagType Char(1)
   , @RedTagUse Varchar(500)
   )
As
    Begin
/*
Template designed by Chris Johnson, Prometic Group September 2015
Stored procedure set out to query multiple databases with the same information and
return it in a collated format
*/
Set NoCount On

        If IsNumeric(@Company) = 0
            Begin
                Select  @Company = Upper(@Company);
            End;

--Red tag

        Declare @RedTagDB Varchar(255)= Db_Name();
        Exec [Process].[UspInsert_RedTagLogs] @StoredProcDb = 'BlackBox' ,
            @StoredProcSchema = 'Report' ,
            @StoredProcName = 'UspResults_Template' ,
            @UsedByType = @RedTagType , @UsedByName = @RedTagUse ,
            @UsedByDb = @RedTagDB;

--list the tables that are to be pulled back from each DB - if they are not found
the script will not be run against that db

        Declare @ListOfTables Varchar(Max) = 'AssetDepreciation,TblApTerms';

--create temporary tables to be pulled from different databases, including a column
to id

        Create Table [#Table1]
```

Author: Johnson, Chris

```
            (
              [DatabaseName] Varchar(150) collate latin1_general_bin
            , [ColumnName] Varchar(500)    collate latin1_general_bin
            );



--create script to pull data from each db into the tables

        Declare @SQL Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            BEGIN
                Insert [#Table1]
                        ( [DatabaseName], [ColumnName] )
                SELECT [t].[DatabaseName]
                     , [t].[ColumnName]
                FROM [#Table1] As [t]
            End
    End';

--Enable this function to check script changes (try to run script directly against
db manually)

--Print @SQL


--execute script against each db, populating the base tables

        Exec [Process].[ExecForEachDB_WithTableCheck] @cmd = @SQL ,
            @SchemaTablesToCheck = @ListOfTables


--define the results you want to return

        Create Table [#Results]
            (
              [DatabaseName] Varchar(150)    collate latin1_general_bin
            , [Results] Varchar(500)    collate latin1_general_bin
            );

--Placeholder to create indexes as required


--script to combine base data and insert into results table

        Insert  [#Results]
                ( [DatabaseName]
                , [Results]
                )
                Select  [DatabaseName]
```

Author: Johnson, Chris

```
                      , [ColumnName]
                From    [#Table1];

Set NoCount Off
--return results

        Select  *
        From    [#Results];

    End;

GO
EXEC sp_addextendedproperty N'MS_Description', N'used to develop new stored procs
(copy and paste code and alter as necessary)', 'SCHEMA', N'Report', 'PROCEDURE',
N'UspResults_Template', NULL, NULL
GO
```

Uses

[Process].[ExecForEachDB_WithTableCheck]
[Process].[UspInsert_RedTagLogs]
[Report]

Author: Johnson, Chris

[Report].[UspResults_TrialBalance]

MS_Description

not in use

Parameters

Name	Data Type	Max Length (Bytes)
@RedTagType	char	1
@RedTagUse	varchar(500)	500

SQL Script

```
CREATE Proc [Report].[UspResults_TrialBalance]
    (
      @RedTagType Char(1)
    , @RedTagUse Varchar(500)
    )
As /*
Template designed by Chris Johnson, Prometic Group April 2016
*/
    Begin
--Red tag

Set NoCount On
        Declare @RedTagDB Varchar(255)= Db_Name();
        Exec [Process].[UspInsert_RedTagLogs] @StoredProcDb = 'BlackBox' ,
            @StoredProcSchema = 'Report' ,
            @StoredProcName = 'UspResults_TrialBalance' ,
            @UsedByType = @RedTagType , @UsedByName = @RedTagUse ,
            @UsedByDb = @RedTagDB;

        Create Table [#Results]
            (
              [Company] Varchar(10)                collate Latin1_General_BIN
            , [ShortName] Varchar(150)             collate Latin1_General_BIN
            , [CompanyName] Varchar(250)         collate Latin1_General_BIN
            , [Currency] Varchar(10)             collate Latin1_General_BIN
            , [GlCode] Varchar(35)                 collate Latin1_General_BIN
            , [GlDescription] Varchar(50)        collate Latin1_General_BIN
            , [ReportIndex1] Varchar(35)         collate Latin1_General_BIN
            , [ReportIndex2] Varchar(35)         collate Latin1_General_BIN
            , [GlGroup] Varchar(10)                 collate Latin1_General_BIN
            , [GlYear] Int
            , [ClosingBalance] Numeric(20 , 2)
            , [Debit] Numeric(20 , 2)
            , [Credit] Numeric(20 , 2)
            , [Period] TinyInt
            , [AccountType] Varchar(250)         collate Latin1_General_BIN
            );
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_Trial-Balance

```
        Insert  [#Results]
                ( [Company]
                , [ShortName]
                , [CompanyName]
                , [Currency]
                , [GlCode]
                , [GlDescription]
                , [ReportIndex1]
                , [ReportIndex2]
                , [GlGroup]
                , [GlYear]
                , [ClosingBalance]
                , [Debit]
                , [Credit]
                , [Period]
                , [AccountType]
                )
                Select  [GH].[Company]
                      , [CN].[ShortName]
                      , [CN].[CompanyName]
                      , [CN].[Currency]
                      , [GH].[GlCode]
                      , [GlDescription] = [GM].[Description]
                      , [ReportIndex1] = Case When [GM].[ReportIndex1] = ''
                                              Then Null
                                              Else [GM].[ReportIndex1]
                                         End
                      , [ReportIndex2] = Case When [GM].[ReportIndex2] = ''
                                              Then Null
                                              Else [GM].[ReportIndex2]
                                         End
                      , [GM].[GlGroup]
                      , [GH].[GlYear]
                      , [GH].[ClosingBalance]
                      , [Debit] = Case When [GH].[ClosingBalance] > 0
                                       Then [GH].[ClosingBalance]
                                       Else 0
                                  End
                      , [Credit] = Case When [GH].[ClosingBalance] < 0
                                        Then [GH].[ClosingBalance]
                                        Else 0
                                   End
                      , [Period] = Case When [GH].[Period] = 'BeginYearBalance'
                                        Then 0
                                        Else Convert(Smallint ,
Replace([GH].[Period] ,
                                                    'ClosingBalPer' ,
                                                    ''))
                                   End
                      , [AccountType] = [GAT].[GLAccountTypeDesc]
                From    [SysproCompany40].[dbo].[GenHistory] Unpivot ( [Closing-
Balance] For [Period] In ( [BeginYearBalance] ,
                                                    [ClosingBalPer1] ,
                                                    [ClosingBalPer2] ,
                                                    [ClosingBalPer3] ,
                                                    [ClosingBalPer4] ,
                                                    [ClosingBalPer5] ,
```

Author: Johnson, Chris

```
                                                                [ClosingBalPer6] ,
                                                                [ClosingBalPer7] ,
                                                                [ClosingBalPer8] ,
                                                                [ClosingBalPer9] ,
                                                                [ClosingBalPer10] ,
                                                                [ClosingBalPer11] ,
                                                                [ClosingBalPer12] ,
                                                                [ClosingBalPer13] ,
                                                                [ClosingBalPer14] ,
                                                                [ClosingBalPer15] ) )
As [GH]
                        Left Join [SysproCompany40].[dbo].[GenMaster] [GM]
                            On [GM].[Company] = [GH].[Company]
                               And [GM].[GlCode] = [GH].[GlCode]
                        Left Join [BlackBox].[Lookups].[CompanyNames] [CN]
                            On [CN].[Company] = [GH].[Company]
                        Left Join [BlackBox].[Lookups].[GLAccountType] [GAT]
                            On [GM].[AccountType] = [GAT].[GLAccountType]
                Where   [GH].[GlCode] Not In ( 'RETAINED' , 'FORCED' );

        Set NoCount Off
        Select  [R].[Company]
              , [R].[ShortName]
              , [R].[CompanyName]
              , [R].[Currency]
              , [R].[GlCode]
              , [R].[GlDescription]
              , [R].[ReportIndex1]
              , [R].[ReportIndex2]
              , [R].[GlGroup]
              , [R].[GlYear]
              , [R].[ClosingBalance]
              , [R].[Debit]
              , [R].[Credit]
              , [R].[Period]
              , [R].[AccountType]
              , [Parse1] = ParseName([R].[GlCode] , 1)
              , [Parse2] = ParseName([R].[GlCode] , 2)
              , [Parse3] = ParseName([R].[GlCode] , 3)
        From    [#Results] [R];

    End;
GO
EXEC sp_addextendedproperty N'MS_Description', N'not in use', 'SCHEMA', N'Report',
'PROCEDURE', N'UspResults_TrialBalance', NULL, NULL
GO
```

Uses

[Lookups].[CompanyNames]
[Lookups].[GLAccountType]
[Process].[UspInsert_RedTagLogs]
[Report]

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_Trial-Balance

Used By

[Report].[UspResults_MovementsTrialBalances]

Author: Johnson, Chris

[Report].[UspResults_UnpaidAssets]

MS_Description

list of unpaid assets

Parameters

Name	Data Type	Max Length (Bytes)
@Company	varchar(max)	max
@RedTagType	char	1
@RedTagUse	varchar(500)	500

SQL Script

```
CREATE Proc [Report].[UspResults_UnpaidAssets]
    (
      @Company Varchar(Max)
    , @RedTagType Char(1)
    , @RedTagUse Varchar(500)
    )
As
    Begin
/*
Template designed by Chris Johnson, Prometic Group September 2015 Stored procedure
set out to query multiple databases with the same information and return it in a
collated format

--exec [Report].[UspResults_UnpaidAssets]  43
*/
        If IsNumeric(@Company) = 0
            Begin
                Select  @Company = Upper(@Company);
            End;


--remove nocount on to speed up query

        Set NoCount On;
--Red tag

        Declare @RedTagDB Varchar(255)= Db_Name();
        Exec [Process].[UspInsert_RedTagLogs] @StoredProcDb = 'BlackBox' ,
            @StoredProcSchema = 'Report' ,
            @StoredProcName = 'UspResults_UnpaidAssets' ,
            @UsedByType = @RedTagType , @UsedByName = @RedTagUse ,
            @UsedByDb = @RedTagDB;
--list the tables that are to be pulled back from each DB - if they are not found
the script will not be run against that db

        Declare @ListOfTables Varchar(Max) = 'ApInvoice,ApJnlSummary,GrnMatching,Grn-
Details,ApControl,ApJnlDistrib';
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-UnpaidAssets

```
--create temporary tables to be pulled from different databases, including a column
to id

        Create Table [#ApInvoice]
            (
              [DatabaseName] Varchar(150)    Collate Latin1_General_BIN
            , [Supplier] Varchar(50)        Collate Latin1_General_BIN
            , [Invoice] Varchar(50)           Collate Latin1_General_BIN
            , [PostCurrency] Varchar(15)    Collate Latin1_General_BIN
            , [ConvRate] Float
            , [MulDiv] Varchar(15)            Collate Latin1_General_BIN
            , [MthInvBal1] Float
            , [MthInvBal2] Float
            , [MthInvBal3] Float
            , [InvoiceYear] Int
            , [InvoiceMonth] Int
            , [JournalDate] DateTime2
            , [InvoiceDate] DateTime2
            );
        Create Table [#ApJnlSummary]
            (
              [DatabaseName] Varchar(150)    Collate Latin1_General_BIN
            , [Supplier] Varchar(50)        Collate Latin1_General_BIN
            , [Invoice] Varchar(50)           Collate Latin1_General_BIN
            , [TrnYear] Int
            , [TrnMonth] Int
            , [Journal] Int
            , [EntryNumber] Int
            );
        Create Table [#GrnMatching]
            (
              [DatabaseName] Varchar(150)    Collate Latin1_General_BIN
            , [Supplier] Varchar(50)        Collate Latin1_General_BIN
            , [Grn] Varchar(35)               Collate Latin1_General_BIN
            , [TransactionType] Varchar(10) Collate Latin1_General_BIN
            , [Journal] Int
            , [EntryNumber] Int
            , [Invoice] Varchar(35)           Collate Latin1_General_BIN
            );
        Create Table [#GrnDetails]
            (
              [DatabaseName] Varchar(150)    Collate Latin1_General_BIN
            , [MatchedValue] Float
            , [DebitRecGlCode] Varchar(35)    Collate Latin1_General_BIN
            , [GrnMonth] Int
            , [GrnYear] Int
            , [Supplier] Varchar(35)        Collate Latin1_General_BIN
            , [Grn] Varchar(35)               Collate Latin1_General_BIN
            , [GrnSource] Varchar(50)        Collate Latin1_General_BIN
            , [Journal] Int
            , [JournalEntry] Int
            );
        Create Table [#ApControl]
            (
              [DatabaseName] Varchar(150) Collate Latin1_General_BIN
            , [FinPeriodDate] DateTime2
```

Author: Johnson, Chris

```
            );
        Create Table [#ApJnlDistrib]
            (
              [DatabaseName] Varchar(150) Collate Latin1_General_BIN
            , [DistrValue] Float
            , [TrnMonth] Int
            , [TrnYear] Int
            , [Journal] Int
            , [EntryNumber] Int
            , [ExpenseGlCode] Varchar(35) Collate Latin1_General_BIN
            );

--create script to pull data from each db into the tables

        Declare @SQL1 Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end'
            + --Only query DBs beginning SysProCompany

            '
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN'
            + --only companies selected in main run, or if companies selected then
all

            '
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT'
            + --count number of tables requested (number of commas plus one)

            '
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')'
            + --Count of the tables requested how many exist in the db

            '
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'','')) '
            + --only if the count matches (all the tables exist in the requested db)
then run the script

            '
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
                Insert [#ApInvoice]
                        ( [DatabaseName]
                        , [Supplier]
                        , [Invoice]
                        , [PostCurrency]
                        , [ConvRate]
                        , [MulDiv]
                        , [MthInvBal1]
```

```
                        , [MthInvBal2]
                        , [MthInvBal3]
                        , [InvoiceYear]
                        , [InvoiceMonth]
                        , [JournalDate]
                        , [InvoiceDate]
                        )
                SELECT [DatabaseName]=@DBCode
                     , [ai].[Supplier]
                     , [ai].[Invoice]
                     , [ai].[PostCurrency]
                     , [ai].[ConvRate]
                     , [ai].[MulDiv]
                     , [ai].[MthInvBal1]
                     , [ai].[MthInvBal2]
                     , [ai].[MthInvBal3]
                     , [ai].[InvoiceYear]
                     , [ai].[InvoiceMonth]
                     , [ai].[JournalDate]
                     , [ai].[InvoiceDate]
                From [ApInvoice] As [ai]
            End
    End';
        Declare @SQL2 Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end'
            + --Only query DBs beginning SysProCompany

            '
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN'
            + --only companies selected in main run, or if companies selected then
all

            '
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT'
            + --count number of tables requested (number of commas plus one)

            '
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')'
            + --Count of the tables requested how many exist in the db

            '
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'','')) '
            + --only if the count matches (all the tables exist in the requested db)
then run the script

            '
            If @ActualCountOfTables=@RequiredCountOfTables
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-UnpaidAssets

```
            BEGIN
            Insert [#ApJnlSummary]
                    ( [DatabaseName]
                    , [Supplier]
                    , [Invoice]
                    , [TrnYear]
                    , [TrnMonth]
                    , [Journal]
                    , [EntryNumber]
                    )
            SELECT [DatabaseName]=@DBCode
                 , [ajs].[Supplier]
                 , [ajs].[Invoice]
                 , [ajs].[TrnYear]
                 , [ajs].[TrnMonth]
                 , [ajs].[Journal]
                 , [ajs].[EntryNumber]
            FROM [ApJnlSummary] As [ajs]
            End
    End';
        Declare @SQL3 Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end'
            + --Only query DBs beginning SysProCompany

            '
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN'
            + --only companies selected in main run, or if companies selected then
all

            '
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT'
            + --count number of tables requested (number of commas plus one)

            '
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')'
            + --Count of the tables requested how many exist in the db

            '
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'','')) '
            + --only if the count matches (all the tables exist in the requested db)
then run the script

            '
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
                Insert [#GrnMatching]
                        ( [DatabaseName]
```

Author: Johnson, Chris

```
                        , [Supplier]
                        , [Grn]
                        , [TransactionType]
                        , [Journal]
                        , [EntryNumber]
                        , [Invoice]
                        )
                SELECT [DatabaseName]=@DBCode
                     , [gm].[Supplier]
                     , [gm].[Grn]
                     , [gm].[TransactionType]
                     , [gm].[Journal]
                     , [gm].[EntryNumber]
                     , [gm].[Invoice]
                FROM [GrnMatching] As [gm]
            End
    End';
        Declare @SQL4 Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end'
            + --Only query DBs beginning SysProCompany

            '
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN'
            + --only companies selected in main run, or if companies selected then
all

            '
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT'
            + --count number of tables requested (number of commas plus one)

            '
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')'
            + --Count of the tables requested how many exist in the db

            '
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'','')) '
            + --only if the count matches (all the tables exist in the requested db)
then run the script

            '
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
                Insert [#GrnDetails]
                        ( [DatabaseName]
                        , [MatchedValue]
                        , [DebitRecGlCode]
                        , [GrnMonth]
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-UnpaidAssets

```
                        , [GrnYear]
                        , [Supplier]
                        , [Grn]
                        , [GrnSource]
                        , [Journal]
                        , [JournalEntry]
                        )
                SELECT [DatabaseName] = @DBCode
                     , [gd].[MatchedValue]
                     , [gd].[DebitRecGlCode]
                     , [gd].[GrnMonth]
                     , [gd].[GrnYear]
                     , [gd].[Supplier]
                     , [gd].[Grn]
                     , [gd].[GrnSource]
                     , [gd].[Journal]
                     , [gd].[JournalEntry]
                FROM [GrnDetails] As [gd]
            End
    End';
        Declare @SQL5 Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end'
            + --Only query DBs beginning SysProCompany

            '
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN'
            + --only companies selected in main run, or if companies selected then
all

            '
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT'
            + --count number of tables requested (number of commas plus one)

            '
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')'
            + --Count of the tables requested how many exist in the db

            '
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'','')) '
            + --only if the count matches (all the tables exist in the requested db)
then run the script

            '
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
                Insert  [#ApControl]
                        ( [DatabaseName]
```

Author: Johnson, Chris

```
                        , [FinPeriodDate]
                        )
                Select [DatabaseName]=@DBCode
                    ,FinPeriodDate = DATEADD(Month, [AC1].[FinPeriod] - 1,
                                        CAST(CAST(FinYear As CHAR(4)) As
DATETIME2))
                From
                    ApControl As AC1
                Where
                    CtlFlag = ''CTL''
            End
    End';
        Declare @SQL6 Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end'
            + --Only query DBs beginning SysProCompany

            '
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN'
            + --only companies selected in main run, or if companies selected then
all

            '
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT'
            + --count number of tables requested (number of commas plus one)

            '
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')'
            + --Count of the tables requested how many exist in the db

            '
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'','')) '
            + --only if the count matches (all the tables exist in the requested db)
then run the script

            '
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
                Insert [#ApJnlDistrib]
                        ( [DatabaseName]
                        , [DistrValue]
                        , [TrnMonth]
                        , [TrnYear]
                        , [Journal]
                        , [EntryNumber]
                        , [ExpenseGlCode]
                        )
                SELECT [DatabaseName]=@DBCode
```

Author: Johnson, Chris

```
                    , [ajd].[DistrValue]
                    , [ajd].[TrnMonth]
                    , [ajd].[TrnYear]
                    , [ajd].[Journal]
                    , [ajd].[EntryNumber]
                    , [ajd].[ExpenseGlCode]
                FROM [ApJnlDistrib] As [ajd]
            End
    End';

--Enable this function to check script changes (try to run script directly against
db manually)

--Print @SQL1


--execute script against each db, populating the base tables

        Exec [Process].[ExecForEachDB] @cmd = @SQL1;
        Exec [Process].[ExecForEachDB] @cmd = @SQL2;
        Exec [Process].[ExecForEachDB] @cmd = @SQL3;
        Exec [Process].[ExecForEachDB] @cmd = @SQL4;
        Exec [Process].[ExecForEachDB] @cmd = @SQL5;
        Exec [Process].[ExecForEachDB] @cmd = @SQL6;

--define the results you want to return


--Placeholder to create indexes as required


--script to combine base data and insert into results table



--return results


--Create AP table

        Select  [ai].[Supplier]
              , [ai].[Invoice]
              , [ai].[PostCurrency]
              , [ai].[ConvRate]
              , [ai].[MulDiv]
              , [ai].[MthInvBal1]
              , [ai].[MthInvBal2]
              , [ai].[MthInvBal3]
              , [DistrValue] = [ajd].[DistrValue]
              , [ExpenseDescription] = [gm].[Description]
              , [ExpenseGlCode] = [ajd].[ExpenseGlCode]
              , [ai].[InvoiceYear]
              , [ai].[InvoiceMonth]
              , [ai].[JournalDate]
              , [ai].[InvoiceDate]
              , [PrevFiscalPeriods] = DateDiff(mm ,
                                               DateAdd(Month ,
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-UnpaidAssets

```
                                                    [ajd].[TrnMonth] - 1 ,
                                                    Cast(Cast([ajd].[TrnYear] As
Char(4)) As DateTime2)) ,
                                                [ApControl].[FinPeriodDate])
        Into    [#AP]
        From    [#ApInvoice] As [ai]
                Left Join [#ApJnlSummary] As [ajs] On [ajs].[Supplier] =
[ai].[Supplier] Collate Latin1_General_BIN
                                                    And [ajs].[Invoice] =
[ai].[Invoice] Collate Latin1_General_BIN
                Left Join [#ApJnlDistrib] As [ajd] On [ajd].[TrnYear] = [ajs].[Trn-
Year]
                                                    And [ajd].[TrnMonth] =
[ajs].[TrnMonth]
                                                    And [ajd].[Journal] =
[ajs].[Journal]
                                                    And [ajd].[EntryNumber] =
[ajs].[EntryNumber]
                Left Join [SysproCompany40]..[GenMaster] As [gm] On [ajd].[ExpenseGl-
Code] Collate Latin1_General_BIN = [gm].[GlCode] Collate Latin1_General_BIN
                Cross Join ( Select [ac].[FinPeriodDate]
                             From   [#ApControl] As [ac]
                           ) As [ApControl]
        Where   ParseName([gm].[GlCode] , 2) In ( '11200' , '11201' , '26001' ,
                                                  '26011' , '26021' , '22999' )
                Or ParseName([gm].[GlCode] , 2) Like '22%%1'
                Or ( ParseName([gm].[GlCode] , 1) = '003'
                     And ParseName([gm].[GlCode] , 2) = '49900'
                   );

--Create GRN table

        Select  [API].[Supplier]
              , [API].[Invoice]
              , [API].[PostCurrency]
              , [API].[ConvRate]
              , [API].[MulDiv]
              , [API].[MthInvBal1]
              , [API].[MthInvBal2]
              , [API].[MthInvBal3]
              , [MatchedValue] = [gd].[MatchedValue]
              , [Description] = [gmst].[Description]
              , [DebitRecGlCode] = [gd].[DebitRecGlCode]
              , [API].[InvoiceYear]
              , [API].[InvoiceMonth]
              , [API].[JournalDate]
              , [API].[InvoiceDate]
              , [PrevFiscalPeriods] = DateDiff(mm ,
                                                DateAdd(Month ,
                                                        [gd].[GrnMonth] - 1 ,
                                                        Cast(Cast([gd].[GrnYear] As
Char(4)) As DateTime2)) ,
                                                [ApControl].[FinPeriodDate])
        Into    [#GRN]
        From    [SysproCompany43]..[ApInvoice] [API]
                Left Join [#GrnMatching] As [gm] On [gm].[Supplier] =
[API].[Supplier] Collate Latin1_General_BIN
                                                  And [gm].[Invoice] =
[API].[Invoice] Collate Latin1_General_BIN
                Left Join [#GrnDetails] As [gd] On [gd].[Supplier] = [gm].[Supplier]
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-UnpaidAssets

```
Collate Latin1_General_BIN
                                                    And [gd].[Grn] = [gm].[Grn]
Collate Latin1_General_BIN
                                                    And [gd].[GrnSource] =
[gm].[TransactionType] Collate Latin1_General_BIN
                                                    And [gd].[Journal] =
[gm].[Journal]
                                                    And [gd].[JournalEntry] =
[gm].[EntryNumber]
                Left Join [SysproCompany40]..[GenMaster] As [gmst] On [gd].[DebitRec-
GlCode] Collate Latin1_General_BIN = [gmst].[GlCode]
                Cross Join ( Select [AC1].[FinPeriodDate]
                             From   [#ApControl] As [AC1]
                           ) As [ApControl]
        Where   ParseName([gmst].[GlCode] , 2) In ( '11200' , '11201' , '26001' ,
                                                    '26011' , '26021' , '22999' )
                Or ParseName([gmst].[GlCode] , 2) Like '22%%1'
                Or ( ParseName([gmst].[GlCode] , 1) = '003'
                     And ParseName([gmst].[GlCode] , 2) = '49900'
                   );

--unpivot both tables and union together

        Select  [t].[Period]
              , [t].[Supplier]
              , [t].[Invoice]
              , [t].[PostCurrency]
              , [t].[ConvRate]
              , [t].[MulDiv]
              , [t].[CompLocalAmt]
              , [t].[MthInvBal]
              , [DistrValue] = Sum([t].[DistrValue])
              , [t].[ExpenseDescription]
              , [t].[ExpenseGlCode]
              , [t].[InvoiceYear]
              , [t].[InvoiceMonth]
              , [t].[JournalDate]
              , [t].[InvoiceDate]
              , [t].[PrevFiscalPeriods]
        From    ( Select    [Period] = 'P' Collate Latin1_General_BIN
                          + Right([MthInvPeriod] , 1)
                        , [Supplier]
                        , [Invoice]
                        , [PostCurrency]
                        , [ConvRate]
                        , [MulDiv]
                        , [MthInvBal]
                        , [CompLocalAmt] = Case When [MulDiv] = 'M' Collate
Latin1_General_BIN
                                                Then [MthInvBal] * [ConvRate]
                                                Else [MthInvBal] / [ConvRate]
                                           End
                        , [DistrValue]
                        , [ExpenseDescription]
                        , [ExpenseGlCode]
                        , [InvoiceYear]
                        , [InvoiceMonth]
                        , [JournalDate] = Convert(Date,[JournalDate])
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-UnpaidAssets

```
                        , [InvoiceDate] = Convert(Date , [InvoiceDate])
                        , [PrevFiscalPeriods]
                From    [#AP] Unpivot ( [MthInvBal] For [MthInvPeriod] In ( [Mth-
InvBal1] ,
                                                        [MthInvBal2] ,
                                                        [MthInvBal3] ) ) As
[ASMT]
                Where   [MthInvBal] <> 0
                Union All
                Select  [Period] = 'P' Collate Latin1_General_BIN
                        + Right([MthInvPeriod] , 1)
                        , [Supplier]
                        , [Invoice]
                        , [PostCurrency]
                        , [ConvRate]
                        , [MulDiv]
                        , [MthInvBal]
                        , [CompLocalAmt] = Case When [MulDiv] = 'M' Collate
Latin1_General_BIN
                                                Then [MthInvBal] * [ConvRate]
                                                Else [MthInvBal] / [ConvRate]
                                           End
                        , [MatchedValue]
                        , [Description]
                        , [DebitRecGlCode]
                        , [InvoiceYear]
                        , [InvoiceMonth]
                        , [JournalDate] = Convert(Date,[JournalDate])
                        , [InvoiceDate] = Convert(Date , [InvoiceDate])
                        , [PrevFiscalPeriods]
                From    [#GRN] Unpivot ( [MthInvBal] For [MthInvPeriod] In (
[MthInvBal1] ,
                                                        [MthInvBal2] ,
                                                        [MthInvBal3] ) ) As
[ASMT]
                Where   [MthInvBal] <> 0
              ) [t]
        Where   [MthInvBal] <> 0
        Group By [t].[Period]
              , [t].[Supplier]
              , [t].[Invoice]
              , [t].[PostCurrency]
              , [t].[ConvRate]
              , [t].[MulDiv]
              , [t].[CompLocalAmt]
              , [t].[MthInvBal]
              , [t].[ExpenseDescription]
              , [t].[ExpenseGlCode]
              , [t].[InvoiceYear]
              , [t].[InvoiceMonth]
              , [t].[JournalDate]
              , [t].[InvoiceDate]
              , [t].[PrevFiscalPeriods];

    End;

GO
EXEC sp_addextendedproperty N'MS_Description', N'list of unpaid assets', 'SCHEMA',
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-UnpaidAssets

```
N'Report', 'PROCEDURE', N'UspResults_UnpaidAssets', NULL, NULL
GO
```

Uses

[Process].[ExecForEachDB]

[Process].[UspInsert_RedTagLogs]

[Report]

Author: Johnson, Chris

[Report].[UspResults_UnpaidGRNAssets]

MS_Description

list of unpaid assets in GRN

Parameters

Name	Data Type	Max Length (Bytes)
@Company	varchar(max)	max
@PeriodYYYYMM	int	4
@RedTagType	char	1
@RedTagUse	varchar(500)	500

SQL Script

```
CREATE Proc [Report].[UspResults_UnpaidGRNAssets]
    (
      @Company Varchar(Max)
    , @PeriodYYYYMM Int
    , @RedTagType Char(1)
    , @RedTagUse Varchar(500)
    )
As
    Begin
/*
Template designed by Chris Johnson, Prometic Group September 2015
///
Stored procedure set out to query multiple databases with the same information and
return it in a collated format        ///
Compares GRN amounts and confirms how much is outstanding
--exec [Report].[UspResults_UnpaidGRNAssets] @Company ='10', @PeriodYYYYMM =201504
*/
        If IsNumeric(@Company) = 0
            Begin
                Select  @Company = Upper(@Company);
            End;


--remove nocount on to speed up query

        Set NoCount On;

--Red tag

        Declare @RedTagDB Varchar(255)= Db_Name();
        Exec [Process].[UspInsert_RedTagLogs] @StoredProcDb = 'BlackBox' ,
            @StoredProcSchema = 'Report' ,
            @StoredProcName = 'UspResults_UnpaidGRNAssets' ,
            @UsedByType = @RedTagType , @UsedByName = @RedTagUse ,
            @UsedByDb = @RedTagDB;
--list the tables that are to be pulled back from each DB - if they are not found
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-UnpaidGRNAssets

```
the script will not be run against that db

        Declare @ListOfTables Varchar(Max) = 'GrnDetails,GrnAdjustment,GrnMatching';

--create temporary tables to be pulled from different databases, including a column
to id

        Create Table [#GrnDets]
            (
              [DatabaseName] Varchar(150)            collate latin1_general_bin
            , [Supplier] Varchar(15)                collate latin1_general_bin
            , [Grn] Varchar(20)                       collate latin1_general_bin
            , [GrnSource] Char(1)                    collate latin1_general_bin
            , [Journal] Int
            , [JournalEntry] Int
            , [DebitRecGlCode] Varchar(35)           collate latin1_general_bin
            , [Description] Varchar(50)               collate latin1_general_bin
            , [OrigGrnValue] Numeric(20 , 8)
            , [PurchaseOrder] Varchar(20)            collate latin1_general_bin
            , [PurchaseOrderLin] Int
            , [StockCode] Varchar(30)                collate latin1_general_bin
            , [StockDescription] Varchar(50)        collate latin1_general_bin
            , [SupCatalogueNum] Varchar(50)           collate latin1_general_bin
            , [Warehouse] Varchar(10)                collate latin1_general_bin
            , [QtyReceived] Numeric(20 , 8)
            , [PostCurrency] Char(3)                collate latin1_general_bin
            , [ConvRate] Numeric(20 , 8)
            , [MulDiv] Char(1)
            , [GrnYear] Int
            , [GrnMonth] Int
            );
        Create Table [#GrnAdjust]
            (
              [DatabaseName] Varchar(150)            collate latin1_general_bin
            , [Supplier] Varchar(15)                collate latin1_general_bin
            , [Grn] Varchar(20)                       collate latin1_general_bin
            , [GrnSource] Char(1)                    collate latin1_general_bin
            , [OrigJournal] Int
            , [OrigJournalEntry] Int
            , [GrnAdjValue] Numeric(20 , 8)
            );
        Create Table [#GrnMatch]
            (
              [DatabaseName] Varchar(150)            collate latin1_general_bin
            , [Supplier] Varchar(15)                collate latin1_general_bin
            , [Grn] Varchar(20)                       collate latin1_general_bin
            , [TransactionType] Char(1)               collate latin1_general_bin
            , [Journal] Int
            , [EntryNumber] Int
            , [MatchedValue] Numeric(20 , 8)
            );

--create script to pull data from each db into the tables

        Declare @SQLGrnDets Varchar(Max) = '
USE [?];
Declare @DB varchar(150),@DBCode varchar(150)
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-UnpaidGRNAssets

```
Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end'
            + --Only query DBs beginning SysProCompany

            '
IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
BEGIN'
            + --only companies selected in main run, or if companies selected then
all

            '
IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables
            + ''', @RequiredCountOfTables INT,@ActualCountOfTables INT'
            + --count number of tables requested (number of commas plus one)

            '
Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_SplitString](@ListOf-
Tables,'','')'
            + --Count of the tables requested how many exist in the db

            '
Select @ActualCountOfTables = COUNT(1) FROM sys.tables
Where name In (Select Value Collate Latin1_General_BIN From BlackBox.dbo.udf_Split-
String(@ListOfTables,'',''))  '
            + --only if the count matches (all the tables exist in the requested db)
then run the script

            '
If @ActualCountOfTables=@RequiredCountOfTables
BEGIN
Insert  [#GrnDets]
(DatabaseName,Supplier,Grn,GrnSource,Journal,JournalEntry,DebitRecGl-
Code,Description,OrigGrnValue,PurchaseOrder,PurchaseOrderLin,StockCode,Stock-
Description,SupCatalogueNum,Warehouse,QtyReceived,PostCurrency,ConvRate,MulDiv,Grn-
Year,GrnMonth)
Select  @DBCode
, Supplier,Grn,GrnSource,Journal,JournalEntry,DebitRecGlCode
, GM2.Description
, OrigGrnValue = Sum(GD.OrigGrnValue)
, PurchaseOrder, PurchaseOrderLin, StockCode, StockDescription, SupCatalogueNum,
Warehouse
, QtyReceived = Sum(GD.QtyReceived)
, PostCurrency, ConvRate, MulDiv, GrnYear, GrnMonth
From    [GrnDetails] [GD] Left Join [SysproCompany40].[dbo].[GenMaster] As [GM2] On
[GD].[DebitRecGlCode] = [GM2].[GlCode]
Where   ([GrnYear]*100)+[GD].[GrnMonth]<=' + Cast(@PeriodYYYYMM As NChar(6))
            + '
And (Substring([GM2].[GlCode],5,5) In
(''11200'',''11201'',''26001'',''26011'',''26021'',''22999'') Or
(Substring([GM2].[GlCode],5,5) Like ''22%%1'' ) Or (Substring([GM2].[GlCode],5,9) =
''49900.003'' ))
And [GM2].[Company] = ''' + @Company
            + '''
Group By [Supplier],[Grn],[DebitRecGlCode],[GM2].[Description],[Purchase-
Order],[PurchaseOrderLin],[StockCode],[StockDescription],[SupCatalogue-
Num],[Warehouse],[GrnSource],[Journal],[JournalEntry],[PostCurrency],[ConvRate],[Mul-
Div],[GrnYear],[GrnMonth];
End
End';
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-UnpaidGRNAssets

```
        Declare @SQLGrnAdjust Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end'
            + --Only query DBs beginning SysProCompany

            '
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN'
            + --only companies selected in main run, or if companies selected then
all

            '
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT'
            + --count number of tables requested (number of commas plus one)

            '
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')'
            + --Count of the tables requested how many exist in the db

            '
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'','')) '
            + --only if the count matches (all the tables exist in the requested db)
then run the script

            '
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
        Insert  [#GrnAdjust]
                ( [DatabaseName]
                , [Supplier]
                , [Grn]
                , [GrnSource]
                , [OrigJournal]
                , [OrigJournalEntry]
                , [GrnAdjValue]
                )
        Select @DBCode
              , [GA].[Supplier]
              , [GA].[Grn]
              , [GA].[GrnSource]
              , [GA].[OrigJournal]
              , [GA].[OrigJournalEntry]
              , [GrnAdjValue] = Sum([GA].[GrnAdjValue])
        From    [GrnAdjustment] [GA] With ( NoLock )
        Where   ( [GA].[AdjYear] * 100 ) + [GA].[AdjMonth] <= '
            + Cast(@PeriodYYYYMM As NChar(6)) + '
        Group By [GA].[Supplier]
              , [GA].[Grn]
              , [GA].[GrnSource]
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-UnpaidGRNAssets

```
              , [GA].[OrigJournal]
              , [GA].[OrigJournalEntry];
            End
    End';
        Declare @SQLGrnMatch Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end'
            + --Only query DBs beginning SysProCompany

            '
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN'
            + --only companies selected in main run, or if companies selected then
all

            '
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
            + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT'
            + --count number of tables requested (number of commas plus one)

            '
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')'
            + --Count of the tables requested how many exist in the db

            '
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'','')) '
            + --only if the count matches (all the tables exist in the requested db)
then run the script

            '
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
        Insert  [#GrnMatch]
        ( [DatabaseName]
        , [Supplier]
        , [Grn]
        , [TransactionType]
        , [Journal]
        , [EntryNumber]
        , [MatchedValue]
        )
        Select @DBCode
              , [GM].[Supplier]
              , [GM].[Grn]
              , [GM].[TransactionType]
              , [GM].[Journal]
              , [GM].[EntryNumber]
              , [MatchedValue] = Sum([GM].[MatchedValue])
        From    [GrnMatching] [GM] With ( NoLock )
        Where   [GM].[MatchedYear] * 100 + [GM].[MatchedMonth] <= '
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-UnpaidGRNAssets

```
            + Cast(@PeriodYYYYMM As NChar(6)) + '
        Group By [GM].[Supplier]
              , [GM].[Grn]
              , [GM].[TransactionType]
              , [GM].[Journal]
              , [GM].[EntryNumber];

            End
    End';

--Enable this function to check script changes (try to run script directly against
db manually)

--Print @SQL


--execute script against each db, populating the base tables

        --Print Len(@SQLGrnDets);

        Exec [Process].[ExecForEachDB] @cmd = @SQLGrnDets;
        --Print @SQLGrnAdjust;

        Exec [Process].[ExecForEachDB] @cmd = @SQLGrnAdjust;
        --Print @SQLGrnMatch;

        Exec [Process].[ExecForEachDB] @cmd = @SQLGrnMatch;

--define the results you want to return

        Create Table [#Results]
            (
              [GrnPeriod] Int
            , [CompanyName] Varchar(150)        collate latin1_general_bin
            , [Supplier] Varchar(15)            collate latin1_general_bin
            , [Grn] Varchar(20)                    collate latin1_general_bin
            , [DebitRecGlCode] Varchar(35)        collate latin1_general_bin
            , [Description] Varchar(50)            collate latin1_general_bin
            , [PurchaseOrder] Varchar(20)        collate latin1_general_bin
            , [PurchaseOrderLine] Int
            , [StockCode] Varchar(30)             collate latin1_general_bin
            , [StockDescription] Varchar(50)    collate latin1_general_bin
            , [SupCatalogueNum] Varchar(50)        collate latin1_general_bin
            , [Warehouse] Varchar(10)            collate latin1_general_bin
            , [QtyReceived] Numeric(20 , 8)
            , [GrnValue] Numeric(20 , 8)
            , [OrigGrnValue] Numeric(20 , 8)
            , [Matched] Numeric(20 , 8)
            , [Adjustments] Numeric(20 , 8)
            , [PostCurrency] Char(3)            collate latin1_general_bin
            , [ConvRate] Numeric(20 , 8)
            , [MultiDiv] Char(1)                collate latin1_general_bin
            , [ShortName] Varchar(250)            collate latin1_general_bin
            );

--Placeholder to create indexes as required
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-UnpaidGRNAssets

```
--script to combine base data and insert into results table

        Insert  [#Results]
                ( [GrnPeriod]
                , [CompanyName]
                , [Supplier]
                , [Grn]
                , [DebitRecGlCode]
                , [Description]
                , [PurchaseOrder]
                , [PurchaseOrderLine]
                , [StockCode]
                , [StockDescription]
                , [SupCatalogueNum]
                , [Warehouse]
                , [QtyReceived]
                , [GrnValue]
                , [OrigGrnValue]
                , [Matched]
                , [Adjustments]
                , [PostCurrency]
                , [ConvRate]
                , [MultiDiv]
                , [ShortName]
                )
                Select  [GrnPeriod] = @PeriodYYYYMM
                      , [CN].[CompanyName]
                      , [GD].[Supplier]
                      , [GD].[Grn]
                      , [GD].[DebitRecGlCode]
                      , [GD].[Description]
                      , [GD].[PurchaseOrder]
                      , [PurchaseOrderLine] = [GD].[PurchaseOrderLin]
                      , [GD].[StockCode]
                      , [GD].[StockDescription]
                      , [GD].[SupCatalogueNum]
                      , [GD].[Warehouse]
                      , [QtyReceived] = Sum([GD].[QtyReceived])
                      , [GrnValue] = Coalesce(Sum([GD].[OrigGrnValue]) , 0)
                        - Coalesce(Sum([GM].[MatchedValue]) , 0)
                        + Coalesce(Sum([GA].[GrnAdjValue]) , 0)
                      , [OrigGrnValue] = Coalesce(Sum([GD].[OrigGrnValue]) , 0)
                      , [Matched] = Sum([GM].[MatchedValue])
                      , [Adjustments] = Sum([GA].[GrnAdjValue])
                      , [GD].[PostCurrency]
                      , [GD].[ConvRate]
                      , [MultiDiv] = [GD].[MulDiv]
                      , [CN].[ShortName]
                From    [#GrnDets] [GD]
                        Left Outer Join [#GrnAdjust] [GA]
                            On [GD].[JournalEntry] = [GA].[OrigJournalEntry]
                               And [GD].[Journal] = [GA].[OrigJournal]
                               And [GD].[GrnSource] = [GA].[GrnSource]
                               And [GD].[Supplier] = [GA].[Supplier]
                               And [GD].[Grn] = [GA].[Grn]
                               And [GA].[DatabaseName] = [GD].[DatabaseName]
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_-UnpaidGRNAssets

```
                        Left Outer Join [#GrnMatch] [GM]
                            On [GD].[JournalEntry] = [GM].[EntryNumber]
                               And [GD].[Journal] = [GM].[Journal]
                               And [GD].[GrnSource] = [GM].[TransactionType]
                               And [GD].[Supplier] = [GM].[Supplier]
                               And [GD].[Grn] = [GM].[Grn]
                               And [GM].[DatabaseName] = [GD].[DatabaseName]
                        Left Join [BlackBox].[Lookups].[CompanyNames] [CN]
                            On [CN].[Company] = [GD].[DatabaseName]
                Group By [GD].[Supplier]
                      , [GD].[Grn]
                      , [GD].[DebitRecGlCode]
                      , [GD].[Description]
                      , [GD].[PurchaseOrder]
                      , [GD].[PurchaseOrderLin]
                      , [GD].[StockCode]
                      , [GD].[StockDescription]
                      , [GD].[SupCatalogueNum]
                      , [GD].[Warehouse]
                      , [GD].[PostCurrency]
                      , [GD].[ConvRate]
                      , [GD].[MulDiv]
                      , [CN].[CompanyName]
                      , [CN].[ShortName]
                Having  ( Sum([OrigGrnValue])
                          - IsNull(Sum(IsNull([GM].[MatchedValue] , 0)) , 0)
                          + Sum(IsNull([GA].[GrnAdjValue] , 0)) <> 0 );

        Drop Table [#GrnAdjust];
        Drop Table [#GrnMatch];
        Drop Table [#GrnDets];

--return results

        Select  [GrnPeriod]
              , [CompanyName]
              , [Supplier]
              , [Grn]
              , [DebitRecGlCode]
              , [Description]
              , [PurchaseOrder]
              , [PurchaseOrderLine]
              , [StockCode]
              , [StockDescription]
              , [SupCatalogueNum]
              , [Warehouse]
              , [QtyReceived]
              , [GrnValue]
              , [OrigGrnValue]
              , [Matched]
              , [Adjustments]
              , [PostCurrency]
              , [ConvRate]
              , [MultiDiv]
              , [ShortName]
        From    [#Results];
```

Author: Johnson, Chris

```
    End;


GO
EXEC sp_addextendedproperty N'MS_Description', N'list of unpaid assets in GRN',
'SCHEMA', N'Report', 'PROCEDURE', N'UspResults_UnpaidGRNAssets', NULL, NULL
GO
```

Uses

[Lookups].[CompanyNames]
[Process].[ExecForEachDB]
[Process].[UspInsert_RedTagLogs]
[Report]

Author: Johnson, Chris

[Report].[UspResults_WipStockRequiredPerJob]

MS_Description

list of stock required for each wip job

Parameters

Name	Data Type	Max Length (Bytes)
@MasterJob	varchar(50)	50
@Company	varchar(10)	10
@RedTagType	char	1
@RedTagUse	varchar(500)	500

SQL Script

```
CREATE Proc [Report].[UspResults_WipStockRequiredPerJob]
    (
      @MasterJob Varchar(50)
    , @Company Varchar(10)
    , @RedTagType Char(1)
    , @RedTagUse Varchar(500)
    )
As /*
Template designed by Chris Johnson, Prometic Group September 2015
Stored procedure set out to query multiple databases with the same information and
return it in a collated format
--Exec [Report].[UspResults_WipStockRequiredPerJob] @MasterJob =94, @Company ='F'
*/
    Set NoCount On;
    If IsNumeric(@Company) = 0
        Begin
            Select  @Company = Upper(@Company);
        End;

--Red tag

    Declare @RedTagDB Varchar(255)= Db_Name();
    Exec [Process].[UspInsert_RedTagLogs] @StoredProcDb = 'BlackBox' ,
        @StoredProcSchema = 'Report' , @StoredProcName = 'UspResults_WipStock-
RequiredPerJob' ,
        @UsedByType = @RedTagType , @UsedByName = @RedTagUse ,
        @UsedByDb = @RedTagDB;


--Convert Job to varchar for querying DB

    Declare @MasterJobVarchar Varchar(20);

--Cater for number jobs
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_Wip-StockRequiredPerJob

```
    Select  @MasterJobVarchar = Case When IsNumeric(@MasterJob) = 1
                                     Then Right('000000000000000' + @MasterJob ,
                                                15)
                                     Else @MasterJob
                                End;

--list the tables that are to be pulled back from each DB - if they are not found
the script will not be run against that db

    Declare @ListOfTables Varchar(Max) = 'WipMasterSub,TblApTerms';

--Set maxmimum recursion to 9000

    Declare @CurrentJobLevel Int = 1
      , @TotalJobLevel Int= 9000
      , @InsertCount Int;

--Create table to capture results

    Create Table [#JobLevelCheck]
        (
          [DatabaseName] Varchar(150)    Collate Latin1_General_BIN
        , [JobLevel] Int
        , [Job] Varchar(20)               Collate Latin1_General_BIN
        , [SubJob] Varchar(20)            Collate Latin1_General_BIN
        );
    Create Table [#WipMasterSub]
        (
          [DatabaseName] Varchar(150)    Collate Latin1_General_BIN
        , [Job] Varchar(20)               Collate Latin1_General_BIN
        , [SubJob] Varchar(20)            Collate Latin1_General_BIN
        );
    Create Table [#WipJobAllMat]
        (
          [DatabaseName] Varchar(150)        Collate Latin1_General_BIN
        , [SequenceNum] Varchar(6)           Collate Latin1_General_BIN
        , [SubJobQty] Numeric(20 , 8)
        , [StockCode] Varchar(30)            Collate Latin1_General_BIN
        , [StockDescription] Varchar(50)    Collate Latin1_General_BIN
        , [UnitQtyReqdEnt] Numeric(20 , 8)
        , [QtyIssuedEnt] Numeric(20 , 8)
        , [FixedQtyPerFlag] Char(1)           Collate Latin1_General_BIN
        , [Uom] Varchar(10)                   Collate Latin1_General_BIN
        , [AllocCompleted] Char(1)           Collate Latin1_General_BIN
        , [OperationOffset] Int
        , [Job] Varchar(20)                   Collate Latin1_General_BIN
        , [ReservedLotSerFlag] Char(1)       Collate Latin1_General_BIN
        , [ReservedLotQty] Numeric(20 , 8)
        );
    Create Table [#WipAllMatLot]
        (
          [DatabaseName] Varchar(150)        Collate Latin1_General_BIN
        , [Job] Varchar(20)                   Collate Latin1_General_BIN
        , [StockCode] Varchar(30)            Collate Latin1_General_BIN
        , [Lot] Varchar(50)                   Collate Latin1_General_BIN
        , [Bin] Varchar(20)                   Collate Latin1_General_BIN
        , [Warehouse] Varchar(20)            Collate Latin1_General_BIN
```

Author: Johnson, Chris

```
        , [QtyReserved] Numeric(20 , 8)
        , [QtyIssued] Numeric(20 , 8)
        );
    Create Table [#WipJobAllLab]
        (
          [DatabaseName] Varchar(150)        Collate Latin1_General_BIN
        , [WorkCentre] Varchar(20)            Collate Latin1_General_BIN
        , [Job] Varchar(20)                    Collate Latin1_General_BIN
        , [Operation] Int
        );
    Create Table [#WipMaster]
        (
          [DatabaseName] Varchar(150)        Collate Latin1_General_BIN
        , [Job] Varchar(20)                    Collate Latin1_General_BIN
        , [QtyToMake] Numeric(20 , 8)
        , [QtyManufactured] Numeric(20 , 8)
        , [JobDescription] Varchar(150)        Collate Latin1_General_BIN
        , [StockCode] Varchar(20)             Collate Latin1_General_BIN
        );
    Create Table [#InvMaster]
        (
          [DatabaseName] Varchar(150)        Collate Latin1_General_BIN
        , [StockCode] Varchar(20)             Collate Latin1_General_BIN
        , [PartCategory] Char(1)             Collate Latin1_General_BIN
        , [IssMultLotsFlag] Char(1)            Collate Latin1_General_BIN
        , [StockUom] Varchar(10)             Collate Latin1_General_BIN
        );
    Create Table [#LotDetail]
        (
          [DatabaseName] Varchar(150)        Collate Latin1_General_BIN
        , [StockCode] Varchar(20)             Collate Latin1_General_BIN
        , [Lot] Varchar(20)                    Collate Latin1_General_BIN
        , [Bin] Varchar(20)                    Collate Latin1_General_BIN
        , [Warehouse] Varchar(30)             Collate Latin1_General_BIN
        , [QtyOnHand] Numeric(20 , 8)
        , [ExpiryDate] DateTime2
        , [CreationDate] DateTime2
        );
    Create Table [#CusLot]
        (
          [DatabaseName] Varchar(150)        Collate Latin1_General_BIN
        , [Lot] Varchar(50)                    Collate Latin1_General_BIN
        , [StockCode] Varchar(30)             Collate Latin1_General_BIN
        , [BleedNumber] Varchar(20)            Collate Latin1_General_BIN
        , [DonorNumber] Varchar(20)            Collate Latin1_General_BIN
        , [VendorBatchNumber] Varchar(50)     Collate Latin1_General_BIN
        , [OldLotNumber] Varchar(20)         Collate Latin1_General_BIN
        , [BleedDate] Varchar(15)             Collate Latin1_General_BIN
        );

--create script to pull data from each db into the tables

    Declare @SQLJobLevelCheck Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_Wip-StockRequiredPerJob

```
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN'
        + --only companies selected in main run, or if companies selected then all

        '
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
        + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT'
        + --count number of tables requested (number of commas plus one)

        '
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')'
        + --Count of the tables requested how many exist in the db

        '
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'','')) '
        + --only if the count matches (all the tables exist in the requested db)
then run the script

        '
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
            Insert  [#JobLevelCheck]
                    ( [DatabaseName]
                    , [JobLevel]
                    , [Job]
                    , [SubJob]
                    )
                    SELECT DatabaseName=@DBCode
                        , JobLevel = 1
                        , [wms].[Job]
                        , [wms].[SubJob]
                    FROM
                        [WipMasterSub]
                        As [wms] With ( NoLock )
                    Where
                        [wms].[Job] = ''' + @MasterJobVarchar + '''
            End
    End';
    Declare @SQLWipMasterSub Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN'
        + --only companies selected in main run, or if companies selected then all

        '
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
        + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables + '''
                    , @RequiredCountOfTables INT
```

Author: Johnson, Chris

```
                    , @ActualCountOfTables INT'
        + --count number of tables requested (number of commas plus one)

        '
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')'
        + --Count of the tables requested how many exist in the db

        '
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'','')) '
        + --only if the count matches (all the tables exist in the requested db)
then run the script

        '
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
                Insert [#WipMasterSub]
                        ( [DatabaseName], [Job], [SubJob] )
                SELECT [DatabaseName]=@DBCode
                     , [wms].[Job]
                     , [wms].[SubJob]
                From [WipMasterSub] As [wms]
            End
    End';
    Declare @SQLWipJobAllMat Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN'
        + --only companies selected in main run, or if companies selected then all

        '
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
        + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT'
        + --count number of tables requested (number of commas plus one)

        '
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')'
        + --Count of the tables requested how many exist in the db

        '
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'','')) '
        + --only if the count matches (all the tables exist in the requested db)
then run the script

        '
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
            Insert [#WipJobAllMat]
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_Wip-StockRequiredPerJob

```
                    ( [DatabaseName]
                    , [SequenceNum]
                    , [StockCode]
                    , [StockDescription]
                    , [UnitQtyReqdEnt]
                    , [QtyIssuedEnt]
                    , [FixedQtyPerFlag]
                    , [Uom]
                    , [AllocCompleted]
                    , [OperationOffset]
                    , [Job]
                    , ReservedLotSerFlag
        , ReservedLotQty
                    )
            SELECT [DatabaseName]=@DBCode
                 , [wjam].[SequenceNum]
                 , [wjam].[StockCode]
                 , [wjam].[StockDescription]
                 , [wjam].[UnitQtyReqdEnt]
                 , [wjam].[QtyIssuedEnt]
                 , [wjam].[FixedQtyPerFlag]
                 , [wjam].[Uom]
                 , [wjam].[AllocCompleted]
                 , [wjam].[OperationOffset]
                 , [wjam].[Job]
                 , ReservedLotSerFlag
        , ReservedLotQty
            From [WipJobAllMat] As [wjam]
            End
    End';
    Declare @SQLWipJobAllLab Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN'
        + --only companies selected in main run, or if companies selected then all

        '
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
        + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT'
        + --count number of tables requested (number of commas plus one)

        '
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')'
        + --Count of the tables requested how many exist in the db

        '
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'',''))  '
        + --only if the count matches (all the tables exist in the requested db)
then run the script
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_Wip-StockRequiredPerJob

```
        '
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
            Insert [#WipJobAllLab]
                    ( [DatabaseName]
                    , [WorkCentre]
                    , [Job]
                    , [Operation]
                    )
            SELECT [DatabaseName]=@DBCode
                 , [wjal].[WorkCentre]
                 , [wjal].[Job]
                 , [wjal].[Operation] FROM [WipJobAllLab] As [wjal]
            End
    End';
    Declare @SQLWipMaster Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN'
        + --only companies selected in main run, or if companies selected then all

        '
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
        + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT'
        + --count number of tables requested (number of commas plus one)

        '
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')'
        + --Count of the tables requested how many exist in the db

        '
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'',''))  '
        + --only if the count matches (all the tables exist in the requested db)
then run the script

        '
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
            Insert [#WipMaster]
                    ( [DatabaseName]
                    , [Job]
                    , [QtyToMake]
                    , [QtyManufactured]
                    , [JobDescription]
                    , StockCode
                    )
            SELECT [DatabaseName]=@DBCode
                 , [Job]
```

Author: Johnson, Chris

```
                , [QtyToMake]
                , [QtyManufactured]
                , [JobDescription]
                , StockCode
            FROM [WipMaster] As [wm]
            End
    End';
    Declare @SQLInvMaster Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN'
        + --only companies selected in main run, or if companies selected then all

        '
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
        + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables + '''
                , @RequiredCountOfTables INT
                , @ActualCountOfTables INT'
        + --count number of tables requested (number of commas plus one)

        '
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')'
        + --Count of the tables requested how many exist in the db

        '
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'','')) '
        + --only if the count matches (all the tables exist in the requested db)
then run the script

        '
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
            Insert [#InvMaster]
                    ( [DatabaseName]
                    , [StockCode]
                    , [PartCategory]
                    , [IssMultLotsFlag]
                    , [StockUom]
                    )
            SELECT [DatabaseName]=@DBCode
                , [StockCode]
                , [PartCategory]
                , [IssMultLotsFlag]
                , [StockUom]
            FROM [InvMaster] As [im]
            End
    End';
    Declare @SQLWipAllMatLot Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_Wip-StockRequiredPerJob

```
Name(),len(db_Name())-13) else null end
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN'
        + --only companies selected in main run, or if companies selected then all

        '
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
        + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT'
        + --count number of tables requested (number of commas plus one)

        '
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')'
        + --Count of the tables requested how many exist in the db

        '
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'','')) '
        + --only if the count matches (all the tables exist in the requested db)
then run the script

        '
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
            Insert [#WipAllMatLot]
                    ( [DatabaseName]
                    , [Job]
                    , [StockCode]
                    , [Lot]
                    , [Bin]
                    , [QtyReserved]
                    , [QtyIssued]
                    , Warehouse
                    )
            SELECT @DBCode
                    , [Job]
                    , [StockCode]
                    , [Lot]
                    , [Bin]
                    , [QtyReserved]
                    , [QtyIssued]
                    , Warehouse
            From [WipAllMatLot] As [waml]
            End
    End';
    Declare @SQLLotDetail Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN'
        + --only companies selected in main run, or if companies selected then all
```

Author: Johnson, Chris

```
        '
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
        + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT'
        + --count number of tables requested (number of commas plus one)

        '
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')'
        + --Count of the tables requested how many exist in the db

        '
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'','')) '
        + --only if the count matches (all the tables exist in the requested db)
then run the script

        '
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
                Insert [#LotDetail]
                        ( [DatabaseName]
                        , [StockCode]
                        , [Lot]
                        , [Bin]
                        , [Warehouse]
                        , [QtyOnHand]
                        , [ExpiryDate]
                        , [CreationDate]
                        )
                Select @DBCode
                  , [ld].[StockCode]
                  , [ld].[Lot]
                  , [ld].[Bin]
                  , [ld].[Warehouse]
                  , [ld].[QtyOnHand]
                  , [ld].[ExpiryDate]
                  , [ld].[CreationDate]
                From
                    [LotDetail] As [ld]
                Where
                    [ld].[QtyOnHand] <> 0
            End
    End';
    Declare @SQLCusLot Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN'
        + --only companies selected in main run, or if companies selected then all

        '
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_Wip-StockRequiredPerJob

```
        + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT'
        + --count number of tables requested (number of commas plus one)

        '
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')'
        + --Count of the tables requested how many exist in the db

        '
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'','')) '
        + --only if the count matches (all the tables exist in the requested db)
then run the script

        '
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
            Declare @SQLSub Varchar(Max) =''Insert  #CusLot
                        ( DatabaseName
                        , Lot
                        , StockCode
                        , BleedNumber
                        , DonorNumber
                        , VendorBatchNumber
                        , OldLotNumber
                        , BleedDate
                        )
                        Select  ''''''+@DBCode+''''''
                                , Lot
                              , StockCode
                              , BleedNumber
                              , DonorNumber
                              , VendorBatchNumber
                              , OldLotNumber
                              , BleedDate
                        From    [dbo].[CusLot+]''
    Exec (@SQLSub)
            End
    End';
    --Print 1

    Exec [Process].[ExecForEachDB] @cmd = @SQLJobLevelCheck;
    --Print 2

    Exec [Process].[ExecForEachDB] @cmd = @SQLWipMasterSub;
    --Print 3

    Exec [Process].[ExecForEachDB] @cmd = @SQLWipJobAllMat;
    --Print 4

    Exec [Process].[ExecForEachDB] @cmd = @SQLWipJobAllLab;
    --Print 5

    Exec [Process].[ExecForEachDB] @cmd = @SQLWipMaster;
```

Author: Johnson, Chris

```
    --Print 6

    Exec [Process].[ExecForEachDB] @cmd = @SQLInvMaster;
    --Print 7

    Exec [Process].[ExecForEachDB] @cmd = @SQLWipAllMatLot;
    --Print 8

    Exec [Process].[ExecForEachDB] @cmd = @SQLLotDetail;
    --Print 9

    Exec [Process].[ExecForEachDB] @cmd = @SQLCusLot;

--iterate through each sub job

    While @CurrentJobLevel < @TotalJobLevel
        Begin
            Insert  [#JobLevelCheck]
                    ( [DatabaseName]
                    , [JobLevel]
                    , [Job]
                    , [SubJob]
                    )
                    Select  [jlc].[DatabaseName]
                          , [JobLevel] = @CurrentJobLevel
                          , [Job] = [jlc].[SubJob]
                          , [wms].[SubJob]
                    From    [#JobLevelCheck] As [jlc]
                            Inner Join [#WipMasterSub] As [wms] On [wms].[Job] =
[jlc].[SubJob]
                                                                   And [jlc].[Database-
Name] = [jlc].[DatabaseName]
                    Where   [jlc].[JobLevel] = @CurrentJobLevel - 1
                            And [wms].[SubJob] <> '';

    --check how many rows added

            Select  @InsertCount = Count(1)
            From    [#JobLevelCheck] As [jlc]
            Where   [jlc].[JobLevel] = @CurrentJobLevel;

    --If row has been added, increase job level by 1 for next iteration

            If @InsertCount > 0
                Begin
                    Select  @CurrentJobLevel = @CurrentJobLevel + 1;
                End;

    --If no rows added, increase job level to 9000 to skip iterations

            If @InsertCount = 0
                Begin
                    Select  @CurrentJobLevel = @TotalJobLevel;
                End;

    --reset insert count for next iteration
```

Author: Johnson, Chris

```
            Select  @InsertCount = 0;
        End;

--Add another row for top level materials without sub jobs

    Insert  [#JobLevelCheck]
            ( [DatabaseName]
            , [JobLevel]
            , [Job]
            , [SubJob]
            )
    Values  ( @Company
            , 1  -- JobLevel - int

            , @MasterJobVarchar  -- Job - varchar(20)

            , @MasterJobVarchar  -- SubJob - varchar(20)

            );


    Select  [MasterJob] = @MasterJob
          , [jlc].[Job]
          , [wjam].[SequenceNum]
          , [jlc].[SubJob]
          , [wjam].[OperationOffset]
          , [SubJobStockCode] = [wm].[StockCode]
          , [SubJobDescription] = [wm].[JobDescription]
          , [SubJobAmount] = [wm].[QtyToMake] - [wm].[QtyManufactured]
          , [SubJobUom] = [im2].[StockUom]
          , [wjam].[StockCode]
          , [wjam].[StockDescription]
          , [wjam].[UnitQtyReqdEnt]
          , [Allocated] = [wjam].[AllocCompleted]
          , [IMPC].[PartCategoryDescription]
          , [im].[PartCategory]
          , [wjam].[QtyIssuedEnt]
          , [wjam].[FixedQtyPerFlag]
          , [wjam].[Uom]
          , [wjal].[WorkCentre]
          , [im].[IssMultLotsFlag]
          , [wjam].[ReservedLotSerFlag]
          , [wjam].[ReservedLotQty]
    From    [#JobLevelCheck] [jlc]
            Left Join [#WipJobAllMat] [wjam] On [jlc].[SubJob] = [wjam].[Job]
                                                And [wjam].[DatabaseName] =
[jlc].[DatabaseName]
            Left Join [#WipJobAllLab] [wjal] On [wjam].[Job] = [wjal].[Job]
                                                And [wjam].[OperationOffset] =
[wjal].[Operation]
            Left Join [#WipMaster] [wm] On [jlc].[SubJob] = [wm].[Job]
                                           And [wm].[DatabaseName] =
[wjal].[DatabaseName]
            Left Join [#InvMaster] [im2] On [im2].[StockCode] = [wm].[StockCode]
                                            And [im2].[DatabaseName] =
[wm].[DatabaseName]
            Left Join [#InvMaster] [im] On [wjam].[StockCode] = [im].[StockCode]
                                           And [im].[DatabaseName] =
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_Wip-StockRequiredPerJob

```
[wjam].[DatabaseName]
            Left Join [Lookups].[InvMaster_PartCategory] [IMPC] On [im].[Part-
Category] = [IMPC].[PartCategoryCode]
    --Where   wjam.AllocCompleted = 'N'

    --        And im.PartCategory <> 'M';

Order By    [jlc].[Job] Asc;


--SELECT * FROM #WipJobAllMat As WJAM

--tidy up

    Drop Table [#JobLevelCheck];
    Drop Table [#WipMasterSub];
    Drop Table [#WipJobAllMat];
    Drop Table [#WipAllMatLot];
    Drop Table [#WipJobAllLab];
    Drop Table [#WipMaster];
    Drop Table [#InvMaster];
    Drop Table [#LotDetail];









GO
EXEC sp_addextendedproperty N'MS_Description', N'list of stock required for each wip
job', 'SCHEMA', N'Report', 'PROCEDURE', N'UspResults_WipStockRequiredPerJob', NULL,
NULL
GO
```

Uses

[Lookups].[InvMaster_PartCategory]
[Process].[ExecForEachDB]
[Process].[UspInsert_RedTagLogs]
[Report]

Author: Johnson, Chris

[Report].[UspResults_WipSubJobStock]

MS_Description

iterates through all jobs and subjobs, returning their stock - used in picklist/bill of materials reports

Parameters

Name	Data Type	Max Length (Bytes)
@MasterJob	varchar(50)	50
@Company	varchar(10)	10
@RedTagType	char	1
@RedTagUse	varchar(500)	500

SQL Script

```
CREATE Proc [Report].[UspResults_WipSubJobStock]
    (
      @MasterJob Varchar(50)
    , @Company Varchar(10)
    , @RedTagType Char(1)
    , @RedTagUse Varchar(500)
    )
As /*
Template designed by Chris Johnson, Prometic Group September 2015
Stored procedure set out to query multiple databases with the same information and
return it in a collated format
--Exec [Report].[UspResults_WipSubJobStock] @MasterJob =94, @Company ='F'
*/
    Set NoCount Off;

    --Cater for lower case company letters being entered

    If IsNumeric(@Company) = 0
        Begin
            Select  @Company = Upper(@Company);
        End;

    --Red tag

    Declare @RedTagDB Varchar(255)= Db_Name();
    Exec [Process].[UspInsert_RedTagLogs] @StoredProcDb = 'BlackBox' ,
        @StoredProcSchema = 'Report' ,
        @StoredProcName = 'UspResults_WipSubJobStock' ,
        @UsedByType = @RedTagType , @UsedByName = @RedTagUse ,
        @UsedByDb = @RedTagDB;

    --Remove any null values generated by crystal

    Delete  From [Process].[Status_WipSubJobStock]
    Where   Coalesce([Job] , '') = '';
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_Wip-SubJobStock

```
    --Remove any incomplete jobs that have been running 10+ minutes

    Delete  From [Process].[Status_WipSubJobStock]
    Where   [IsComplete] = 0
            And DateDiff(Minute , [StartTime] , GetDate()) > 10;

    --Remove data reported more than a day ago

    Delete  From [Report].[Results_WipSubJobStock]
    Where   DateDiff(Day , [StartTime] , GetDate()) > 1;

    Declare @StartTime DateTime2 = GetDate()
      , @CompleteTime DateTime2
      , @LatestStartTime DateTime2;

    --Work out if proc has been run in the past three minutes and has not completed
yet

    Select  @LatestStartTime = Max([s].[StartTime])
    From    [Process].[Status_WipSubJobStock] [s]
    Where   ( [s].[IsComplete] = 0
              Or DateDiff(Minute , [s].[CompleteTime] , GetDate()) < 3
            )
            And [s].[Job] = @MasterJob
            And [s].[Company] = @Company;

    -- if the proc has not been run in the past three minutes start capturing data

    If @LatestStartTime Is Null
        Begin
            Insert  [Process].[Status_WipSubJobStock]
                    ( [StartTime]
                    , [Job]
                    , [Company]
                    )
                    Select  @StartTime
                          , @MasterJob
                          , @Company;

--Convert Job to varchar for querying DB

            Declare @MasterJobVarchar Varchar(20);
            --= RIGHT('000000000000000' + CAST(@MasterJob As VARCHAR(20)),15);


--Cater for number jobs

            Select  @MasterJobVarchar = Case When IsNumeric(@MasterJob) = 1
                                             Then Right('000000000000000'
                                                        + @MasterJob , 15)
                                             Else @MasterJob
                                        End;

--list the tables that are to be pulled back from each DB - if they are not found
the script will not be run against that db
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_Wip-SubJobStock

```
            Declare @ListOfTables Varchar(Max) = 'WipMasterSub,TblApTerms';

--Set maxmimum recursion to 9000

            Declare @CurrentJobLevel Int = 1
              , @TotalJobLevel Int= 9000
              , @InsertCount Int;

--Create tables to capture results

            Create Table [#JobLevelCheck]
                (
                  [DatabaseName] Varchar(150) Collate Latin1_General_BIN
                , [JobLevel] Int
                , [Job] Varchar(20) Collate Latin1_General_BIN
                , [SubJob] Varchar(20)
                    Collate Latin1_General_BIN
                    Constraint [JobLevelCheck_AllKeys]
                    Primary Key NonClustered
                    ( [DatabaseName] , [Job] , [SubJob] )
                    With ( Ignore_Dup_Key = On )
                );
            Create Table [#WipMasterSub]
                (
                  [DatabaseName] Varchar(150) Collate Latin1_General_BIN
                , [Job] Varchar(20) Collate Latin1_General_BIN
                , [SubJob] Varchar(20) Collate Latin1_General_BIN
                );
            Create Table [#WipJobAllMat]
                (
                  [DatabaseName] Varchar(150) Collate Latin1_General_BIN
                , [SequenceNum] Varchar(6) Collate Latin1_General_BIN
                , [SubJobQty] Numeric(20 , 8)
                , [StockCode] Varchar(30) Collate Latin1_General_BIN
                , [StockDescription] Varchar(50) Collate Latin1_General_BIN
                , [UnitQtyReqdEnt] Numeric(20 , 8)
                , [QtyIssuedEnt] Numeric(20 , 8)
                , [FixedQtyPerFlag] Char(1) Collate Latin1_General_BIN
                , [Uom] Varchar(10) Collate Latin1_General_BIN
                , [AllocCompleted] Char(1) Collate Latin1_General_BIN
                , [OperationOffset] Int
                , [Job] Varchar(20) Collate Latin1_General_BIN
                , [ReservedLotSerFlag] Char(1) Collate Latin1_General_BIN
                , [ReservedLotQty] Numeric(20 , 8)
                );
            Create Table [#WipAllMatLot]
                (
                  [DatabaseName] Varchar(150) Collate Latin1_General_BIN
                , [Job] Varchar(20) Collate Latin1_General_BIN
                , [StockCode] Varchar(30) Collate Latin1_General_BIN
                , [Lot] Varchar(50) Collate Latin1_General_BIN
                , [Bin] Varchar(20) Collate Latin1_General_BIN
                , [Warehouse] Varchar(20) Collate Latin1_General_BIN
                , [QtyReserved] Numeric(20 , 8)
                , [QtyIssued] Numeric(20 , 8)
                );
            Create Table [#WipJobAllLab]
```

Author: Johnson, Chris

```
                (
                  [DatabaseName] Varchar(150) Collate Latin1_General_BIN
                , [WorkCentre] Varchar(20) Collate Latin1_General_BIN
                , [Job] Varchar(20) Collate Latin1_General_BIN
                , [Operation] Int
                );
            Create Table [#WipMaster]
                (
                  [DatabaseName] Varchar(150) Collate Latin1_General_BIN
                , [Job] Varchar(20) Collate Latin1_General_BIN
                , [QtyToMake] Numeric(20 , 8)
                , [QtyManufactured] Numeric(20 , 8)
                , [JobDescription] Varchar(150)
                , [StockCode] Varchar(20) Collate Latin1_General_BIN
                );
            Create Table [#InvMaster]
                (
                  [DatabaseName] Varchar(150) Collate Latin1_General_BIN
                , [StockCode] Varchar(20) Collate Latin1_General_BIN
                , [PartCategory] Char(1) Collate Latin1_General_BIN
                , [IssMultLotsFlag] Char(1) Collate Latin1_General_BIN
                , [StockUom] Varchar(10) Collate Latin1_General_BIN
                , [Decimals] Int
                );
            Create Table [#LotDetail]
                (
                  [DatabaseName] Varchar(150) Collate Latin1_General_BIN
                , [StockCode] Varchar(20) Collate Latin1_General_BIN
                , [Lot] Varchar(20) Collate Latin1_General_BIN
                , [Bin] Varchar(20) Collate Latin1_General_BIN
                , [Warehouse] Varchar(30) Collate Latin1_General_BIN
                , [QtyOnHand] Numeric(20 , 8)
                , [ExpiryDate] DateTime2
                , [CreationDate] DateTime2
                );
            Create Table [#CusLot]
                (
                  [DatabaseName] Varchar(150) Collate Latin1_General_BIN
                , [Lot] Varchar(50) Collate Latin1_General_BIN
                , [StockCode] Varchar(30) Collate Latin1_General_BIN
                , [BleedNumber] Varchar(20) Collate Latin1_General_BIN
                , [DonorNumber] Varchar(20) Collate Latin1_General_BIN
                , [VendorBatchNumber] Varchar(50) Collate Latin1_General_BIN
                , [OldLotNumber] Varchar(20) Collate Latin1_General_BIN
                , [BleedDate] Varchar(15) Collate Latin1_General_BIN
                );

--create script to pull data from each db into the tables


-- all jobs that attach to a high level

            Declare @SQLJobLevelCheck Varchar(Max) = '    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end'
                + --Only query DBs beginning SysProCompany
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_Wip-SubJobStock

```
                '
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN'
                + --only companies selected in main run, or if companies selected
then all

                '
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
                + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT'
                + --count number of tables requested (number of commas plus one)

                '
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')'
                + --Count of the tables requested how many exist in the db

                '
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'','')) '
                + --only if the count matches (all the tables exist in the requested
db) then run the script

                '
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
            Insert  [#JobLevelCheck]
                    ( [DatabaseName]
                    , [JobLevel]
                    , [Job]
                    , [SubJob]
                    )
                    SELECT DatabaseName=@DBCode
                        , JobLevel = 1
                        , [wms].[Job]
                        , [wms].[SubJob]
                    FROM
                        [WipMasterSub]
                        As [wms] With ( NoLock )
                    Where
                        [wms].[Job] = ''' + @MasterJobVarchar + '''
            End
    End';

-- All sub jobs and their masters

            Declare @SQLWipMasterSub Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end'
                + --Only query DBs beginning SysProCompany

                '
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_Wip-SubJobStock

```
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN'
                + --only companies selected in main run, or if companies selected
then all

                '
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
                + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT'
                + --count number of tables requested (number of commas plus one)

                '
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')'
                + --Count of the tables requested how many exist in the db

                '
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'','')) '
                + --only if the count matches (all the tables exist in the requested
db) then run the script

                '
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
                Insert [#WipMasterSub]
                        ( [DatabaseName], [Job], [SubJob] )
                SELECT [DatabaseName]=@DBCode
                     , [wms].[Job]
                     , [wms].[SubJob]
                From [WipMasterSub] As [wms]
            End
    End';

-- list of all materials required

            Declare @SQLWipJobAllMat Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end'
                + --Only query DBs beginning SysProCompany

                '
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN'
                + --only companies selected in main run, or if companies selected
then all

                '
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
                + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT'
```

Author: Johnson, Chris

```
                + --count number of tables requested (number of commas plus one)

                '
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')'
                + --Count of the tables requested how many exist in the db

                '
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'','')) '
                + --only if the count matches (all the tables exist in the requested
db) then run the script

                '
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
            Insert [#WipJobAllMat]
                    ( [DatabaseName]
                    , [SequenceNum]
                    , [StockCode]
                    , [StockDescription]
                    , [UnitQtyReqdEnt]
                    , [QtyIssuedEnt]
                    , [FixedQtyPerFlag]
                    , [Uom]
                    , [AllocCompleted]
                    , [OperationOffset]
                    , [Job]
                    )
            SELECT [DatabaseName]=@DBCode
                 , [wjam].[SequenceNum]
                 , [wjam].[StockCode]
                 , [wjam].[StockDescription]
                 , [wjam].[UnitQtyReqdEnt]
                 , [wjam].[QtyIssuedEnt]
                 , [wjam].[FixedQtyPerFlag]
                 , [wjam].[Uom]
                 , [wjam].[AllocCompleted]
                 , [wjam].[OperationOffset]
                 , [wjam].[Job]
            From [WipJobAllMat] As [wjam]
            End
    End';

-- list of all jobs and operations

            Declare @SQLWipJobAllLab Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end'
                + --Only query DBs beginning SysProCompany

                '
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN'
                + --only companies selected in main run, or if companies selected
```

```
then all

                '
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
                + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT'
                + --count number of tables requested (number of commas plus one)

                '
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')'
                + --Count of the tables requested how many exist in the db

                '
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'','')) '
                + --only if the count matches (all the tables exist in the requested
db) then run the script

                '
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
            Insert [#WipJobAllLab]
                    ( [DatabaseName]
                    , [WorkCentre]
                    , [Job]
                    , [Operation]
                    )
            SELECT [DatabaseName]=@DBCode
                 , [wjal].[WorkCentre]
                 , [wjal].[Job]
                 , [wjal].[Operation] FROM [WipJobAllLab] As [wjal]
            End
    End';

-- list of materials that are to be made

            Declare @SQLWipMaster Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end'
                + --Only query DBs beginning SysProCompany

                '
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN'
                + --only companies selected in main run, or if companies selected
then all

                '
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
                + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables + '''
                    , @RequiredCountOfTables INT
```

Author: Johnson, Chris

```
                    , @ActualCountOfTables INT'
                + --count number of tables requested (number of commas plus one)

                '
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')'
                + --Count of the tables requested how many exist in the db

                '
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'',')) '
                + --only if the count matches (all the tables exist in the requested
db) then run the script

                '
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
            Insert [#WipMaster]
                    ( [DatabaseName]
                    , [Job]
                    , [QtyToMake]
                    , [QtyManufactured]
                    , [JobDescription]
                    , StockCode
                    )
            SELECT [DatabaseName]=@DBCode
                 , [Job]
                 , [QtyToMake]
                 , [QtyManufactured]
                 , [JobDescription]
                 , StockCode
            FROM [WipMaster] As [wm]
            End
    End';

-- list of all possible stock

            Declare @SQLInvMaster Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end'
                + --Only query DBs beginning SysProCompany

                '
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN'
                + --only companies selected in main run, or if companies selected
then all

                '
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
                + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT'
                + --count number of tables requested (number of commas plus one)
```

```
                '
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')'
                + --Count of the tables requested how many exist in the db

                '
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'','')) '
                + --only if the count matches (all the tables exist in the requested
db) then run the script

                '
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
            Insert [#InvMaster]
                    ( [DatabaseName]
                    , [StockCode]
                    , [PartCategory]
                    , [IssMultLotsFlag]
                    , [StockUom]
                    , [Decimals]
                    )
            SELECT [DatabaseName]=@DBCode
                 , [StockCode]
                 , [PartCategory]
                 , [IssMultLotsFlag]
                 , [StockUom]
                 , [Decimals]
            FROM [InvMaster] As [im]
            End
    End';

-- details of all reserved/allocated materials required

            Declare @SQLWipAllMatLot Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end'
                + --Only query DBs beginning SysProCompany

                '
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN'
                + --only companies selected in main run, or if companies selected
then all

                '
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
                + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT'
                + --count number of tables requested (number of commas plus one)

                '
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_Wip-SubJobStock

```
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')'
                + --Count of the tables requested how many exist in the db

                '
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'','')) '
                + --only if the count matches (all the tables exist in the requested
db) then run the script

                '
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
            Insert [#WipAllMatLot]
                    ( [DatabaseName]
                    , [Job]
                    , [StockCode]
                    , [Lot]
                    , [Bin]
                    , [QtyReserved]
                    , [QtyIssued]
                    , Warehouse
                    )
            SELECT @DBCode
                    , [Job]
                    , [StockCode]
                    , [Lot]
                    , [Bin]
                    , [QtyReserved]
                    , [QtyIssued]
                    , Warehouse
            From [WipAllMatLot] As [waml]
            End
    End';

-- details of all lots

            Declare @SQLLotDetail Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end'
                + --Only query DBs beginning SysProCompany

                '
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN'
                + --only companies selected in main run, or if companies selected
then all

                '
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
                + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT'
                + --count number of tables requested (number of commas plus one)
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_Wip-SubJobStock

```
                '
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')'
                + --Count of the tables requested how many exist in the db

                '
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'','')) '
                + --only if the count matches (all the tables exist in the requested
db) then run the script

                '
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
                Insert [#LotDetail]
                        ( [DatabaseName]
                        , [StockCode]
                        , [Lot]
                        , [Bin]
                        , [Warehouse]
                        , [QtyOnHand]
                        , [ExpiryDate]
                        , [CreationDate]
                        )
                Select @DBCode
                  , [ld].[StockCode]
                  , [ld].[Lot]
                  , [ld].[Bin]
                  , [ld].[Warehouse]
                  , [ld].[QtyOnHand]
                  , [ld].[ExpiryDate]
                  , [ld].[CreationDate]
                From
                    [LotDetail] As [ld]
                Where
                    [ld].[QtyOnHand] <> 0
            End
    End';

-- details about reserved lots from custom lot fields

            Declare @SQLCusLot Varchar(Max) = '
    USE [?];
    Declare @DB varchar(150),@DBCode varchar(150)
    Select @DB = DB_NAME(),@DBCode = case when len(db_Name())>13 then right(db_-
Name(),len(db_Name())-13) else null end'
                + --Only query DBs beginning SysProCompany

                '
    IF left(@DB,13)=''SysproCompany'' and right(@DB,3)<>''SRS''
    BEGIN'
                + --only companies selected in main run, or if companies selected
then all

                '
        IF @DBCode in (''' + Replace(@Company , ',' , ''',''') + ''') or '''
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_Wip-SubJobStock

```
                + Upper(@Company) + ''' = ''ALL''
            Declare @ListOfTables VARCHAR(max) = ''' + @ListOfTables + '''
                    , @RequiredCountOfTables INT
                    , @ActualCountOfTables INT'
                + --count number of tables requested (number of commas plus one)

                '
            Select @RequiredCountOfTables= count(1) from  BlackBox.dbo.[udf_Split-
String](@ListOfTables,'','')'
                + --Count of the tables requested how many exist in the db

                '
            Select @ActualCountOfTables = COUNT(1) FROM sys.tables
            Where name In (Select Value Collate Latin1_General_BIN From Black-
Box.dbo.udf_SplitString(@ListOfTables,'','')) '
                + --only if the count matches (all the tables exist in the requested
db) then run the script

                '
            If @ActualCountOfTables=@RequiredCountOfTables
            BEGIN
            Declare @SQLSub Varchar(Max) =''Insert  #CusLot
                        ( DatabaseName
                        , Lot
                        , StockCode
                        , BleedNumber
                        , DonorNumber
                        , VendorBatchNumber
                        , OldLotNumber
                        , BleedDate
                        )
                        Select  ''''''+@DBCode+''''''
                                , Lot
                              , StockCode
                              , BleedNumber
                              , DonorNumber
                              , VendorBatchNumber
                              , OldLotNumber
                              , BleedDate
                        From    [dbo].[CusLot+]''
    Exec (@SQLSub)
            End
    End';
    --Print 1

            Exec [sys].[sp_MSforeachdb] @SQLJobLevelCheck;
    --Print 2

            Exec [sys].[sp_MSforeachdb] @SQLWipMasterSub;
    --Print 3

            Exec [sys].[sp_MSforeachdb] @SQLWipJobAllMat;
    --Print 4

            Exec [sys].[sp_MSforeachdb] @SQLWipJobAllLab;
    --Print 5

            Exec [sys].[sp_MSforeachdb] @SQLWipMaster;
```

Author: Johnson, Chris

```
    --Print 6

            Exec [sys].[sp_MSforeachdb] @SQLInvMaster;
    --Print 7

            Exec [sys].[sp_MSforeachdb] @SQLWipAllMatLot;
    --Print 8

            Exec [sys].[sp_MSforeachdb] @SQLLotDetail;
    --Print 9

            Exec [sys].[sp_MSforeachdb] @SQLCusLot;

--iterate through each sub job

            While @CurrentJobLevel < @TotalJobLevel --Total Job level defined at
beginning of proc, currently 9000 iterations (CJ 2016-01-13)

                Begin
                    Insert  [#JobLevelCheck]
                            ( [DatabaseName]
                            , [JobLevel]
                            , [Job]
                            , [SubJob]
                            )
                            Select  [jlc].[DatabaseName]
                                  , [JobLevel] = @CurrentJobLevel
                                  , [Job] = [jlc].[SubJob]
                                  , [wms].[SubJob]
                            From    [#JobLevelCheck] As [jlc]
                                    Inner Join [#WipMasterSub] As [wms]
                                        On [wms].[Job] = [jlc].[SubJob]
                                           And [jlc].[DatabaseName] =
[jlc].[DatabaseName]
                            Where   [jlc].[JobLevel] = @CurrentJobLevel - 1
                                    And [wms].[SubJob] <> '';

    --check how many rows added

                    Select  @InsertCount = Count(1)
                    From    [#JobLevelCheck] As [jlc]
                    Where   [jlc].[JobLevel] = @CurrentJobLevel;

    --If row has been added, increase job level by 1 for next iteration

                    If @InsertCount > 0
                        Begin
                            Select  @CurrentJobLevel = @CurrentJobLevel + 1;
                        End;

    --If no rows added, increase job level to 9000 to skip iterations

                    If @InsertCount = 0
                        Begin
                            Select  @CurrentJobLevel = @TotalJobLevel;
                        End;
```

Author: Johnson, Chris

```
    --reset insert count for next iteration

                    Select  @InsertCount = 0;
                End;

--Add another row for top level materials without sub jobs

            Insert  [#JobLevelCheck]
                    ( [DatabaseName]
                    , [JobLevel]
                    , [Job]
                    , [SubJob]
                    )
            Values  ( @Company
                    , 1  -- JobLevel - int

                    , @MasterJobVarchar  -- Job - varchar(20)

                    , @MasterJobVarchar  -- SubJob - varchar(20)

                    );


            Insert  [Report].[Results_WipSubJobStock]
                    ( [StartTime]
                    , [Job]
                    , [SubJob]
                    , [SubJobDescription]
                    , [SubJobUom]
                    , [SequenceNum]
                    , [SubJobQty]
                    , [StockCode]
                    , [StockDescription]
                    , [UnitQtyReqdEnt]
                    , [QtyIssuedEnt]
                    , [FixedQtyPerFlag]
                    , [Uom]
                    , [AllocCompleted]
                    , [OperationOffset]
                    , [WorkCentre]
                    , [SubJobQtyTotal]
                    , [IssMultLotsFlag]
                    , [ReservedLotSerFlag]
                    , [ReservedLotQty]
                    , [ReservedLot]
                    , [ReservedLotBin]
                    , [ReservedLotWarehouse]
                    , [ReservedLotQtyReserved]
                    , [ReservedLotQtyIssued]
                    , [AvailableLot]
                    , [AvailableLotBin]
                    , [AvailableLotWarehouse]
                    , [AvailableLotQtyOnHand]
                    , [AvailableLotExpiryDate]
                    , [AvailableLotCreationDate]
                    , [ReservedLotBleedNumber]
                    , [ReservedLotDonorNumber]
```

Author: Johnson, Chris

```
                    , [ReservedLotVendorBatchNumber]
                    , [ReservedLotOldLotNumber]
                    , [ReservedLotBleedDate]
                    )
                    Select  @StartTime
                          , [jlc].[Job]
                          , [jlc].[SubJob]
                          , [SubJobDescription] = [wm].[JobDescription]
                          , [SubJobUom] = [im2].[StockUom]
                          , [wjam].[SequenceNum]
                          , [SubJobQty] = [wm].[QtyToMake]
                            - [wm].[QtyManufactured]
                          , [wjam].[StockCode]
                          , [wjam].[StockDescription]
                          , [wjam].[UnitQtyReqdEnt]
                          , [wjam].[QtyIssuedEnt]
                          , [wjam].[FixedQtyPerFlag]
                          , [wjam].[Uom]
                          , [wjam].[AllocCompleted]
                          , [wjam].[OperationOffset]
                          , [wjal].[WorkCentre]
                          , [SubJobQtyTotal] = Case When [wjam].[FixedQtyPerFlag] =
'N'
                                                    Then ( ( [wm].[QtyToMake]
                                                             - [wm].[Qty-
Manufactured] )
                                                           * [wjam].[UnitQtyReqdEnt]
)
                                                    Else [wjam].[UnitQtyReqdEnt]
                                               End
                          , [im].[IssMultLotsFlag]
                          , [wjam].[ReservedLotSerFlag]
                          , [wjam].[ReservedLotQty]
                          , [ReservedLot] = [waml].[Lot]
                          , [ReservedLotBin] = [waml].[Bin]
                          , [ReservedLotWarehouse] = [waml].[Warehouse]
                          , [ReservedLotQtyReserved] = [waml].[QtyReserved]
                          , [ReservedLotQtyIssued] = [waml].[QtyIssued]
                          , [AvailableLot] = [ld].[Lot]
                          , [AvailableLotBin] = [ld].[Bin]
                          , [AvailableLotWarehouse] = [ld].[Warehouse]
                          , [AvailableLotQtyOnHand] = [ld].[QtyOnHand]
                          , [AvailableLotExpiryDate] = [ld].[ExpiryDate]
                          , [AvailableLotCreationDate] = [ld].[CreationDate]
                          , [ReservedLotBleedNumber] = [CL].[BleedNumber]
                          , [ReservedLotDonorNumber] = [CL].[DonorNumber]
                          , [ReservedLotVendorBatchNumber] = [CL].[VendorBatch-
Number]
                          , [ReservedLotOldLotNumber] = [CL].[OldLotNumber]
                          , [ReservedLotBleedDate] = [CL].[BleedDate]
                    From    [#JobLevelCheck] [jlc]
                            Left Join [#WipJobAllMat] [wjam]
                                On [jlc].[SubJob] = [wjam].[Job]
                                   And [wjam].[DatabaseName] = [jlc].[DatabaseName]
                            Left Join [#WipJobAllLab] [wjal]
                                On [wjam].[Job] = [wjal].[Job]
                                   And [wjam].[OperationOffset] = [wjal].[Operation]
                                   And [wjal].[DatabaseName] = [wjam].[DatabaseName]
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_Wip-SubJobStock

```
                            Left Join [#WipMaster] [wm]
                                On [jlc].[SubJob] = [wm].[Job]
                                   And [wm].[DatabaseName] = [jlc].[DatabaseName]
                            Left Join [#InvMaster] As [im2]
                                On [im2].[StockCode] = [wm].[StockCode]
                                   And [im2].[DatabaseName] = [wm].[DatabaseName]
                            Left Join [#InvMaster] [im]
                                On [wjam].[StockCode] = [im].[StockCode]
                                   And [im].[DatabaseName] = [wjam].[DatabaseName]
                            Left Join [#LotDetail] As [ld]
                                On [ld].[StockCode] = [im].[StockCode]
                                   And [im].[IssMultLotsFlag] = 'Y'
                                   And Coalesce([wjam].[ReservedLotSerFlag] ,
                                                'N') <> 'Y'
                                   And [ld].[DatabaseName] = [im].[DatabaseName]
                            Left Join [#WipAllMatLot] As [waml]
                                On [waml].[Job] = [wjam].[Job]
                                   And [waml].[StockCode] = [wjam].[StockCode]
                                   And [waml].[DatabaseName] = [wjam].[DatabaseName]
                            Left Join [#CusLot] As [CL]
                                On [CL].[Lot] = [waml].[Lot]
                                   And [CL].[StockCode] = [waml].[StockCode]
                                   And [CL].[DatabaseName] = [waml].[DatabaseName]
                    Where   --wjam.AllocCompleted = 'N'

                        --And

                            [im].[PartCategory] <> 'M';

            Update  [Process].[Status_WipSubJobStock]
            Set     [CompleteTime] = GetDate()
                  , [IsComplete] = 1
            Where   [StartTime] = @StartTime
                    And [Job] = @MasterJob
                    And [Company] = @Company;
--tidy up

            Drop Table [#JobLevelCheck];
            Drop Table [#WipMasterSub];
            Drop Table [#WipJobAllMat];
            Drop Table [#WipAllMatLot];
            Drop Table [#WipJobAllLab];
            Drop Table [#WipMaster];
            Drop Table [#InvMaster];
            Drop Table [#LotDetail];
        End;

--Set StartTime to last start date

    If @LatestStartTime Is Not Null
        Begin
            Select  @StartTime = @LatestStartTime;
        End;

--Hold Process until Results are ready

    Declare @Complete Bit = 0;
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults_Wip-SubJobStock

```
    While @Complete < 1
        Begin
            Select  @Complete = [swsjs].[IsComplete]
            From    [Process].[Status_WipSubJobStock] As [swsjs]
            Where   [swsjs].[StartTime] = @StartTime;
            WaitFor Delay '00:00:01';
        End;

--Return Results

    Create Table [#ReservedLotKeys]
        (
          [StartTime] DateTime2
        , [StockCode] Varchar(20) Collate Latin1_General_BIN
        , [ReservedLots] Bit
        );
    Insert  [#ReservedLotKeys]
            ( [StartTime]
            , [StockCode]
            , [ReservedLots]
            )
            Select  [RWSJS].[StartTime]
                  , [RWSJS].[StockCode]
                  , [ReservedLots] = Case When Max([RWSJS].[ReservedLot]) Is Not
Null
                                          Then 1
                                          Else 0
                                     End
            From    [Report].[Results_WipSubJobStock] As [RWSJS]
            Group By [RWSJS].[StockCode]
                  , [RWSJS].[StartTime];

    Select  [MasterJob] = @MasterJob
          , [Job] = Case When IsNumeric([r].[Job]) = 1
                         Then Cast(Cast([r].[Job] As Int) As Varchar(20))
                         Else [r].[Job]
                    End
          , [SubJob] = Case When IsNumeric([r].[SubJob]) = 1
                            Then Cast(Cast([r].[SubJob] As Int) As Varchar(20))
                            Else [r].[SubJob]
                       End
          , [r].[SubJobDescription]
          , [SequenceNum] = Case When IsNumeric([r].[SequenceNum]) = 1
                                 Then Cast(Cast([r].[SequenceNum] As Int) As
Varchar(20))
                                 Else [r].[SequenceNum]
                            End
          , [r].[SubJobQty]
          , [StockCode] = Case When IsNumeric([r].[StockCode]) = 1
                               Then Cast(Cast([r].[StockCode] As Int) As
Varchar(20))
                               Else [r].[StockCode]
                          End
          , [r].[StockDescription]
          , [r].[UnitQtyReqdEnt]
          , [r].[QtyIssuedEnt]
          , [r].[FixedQtyPerFlag]
```

Author: Johnson, Chris

```
          , [Uom] = Upper([r].[Uom])
          , [r].[AllocCompleted]
          , [r].[OperationOffset]
          , [r].[WorkCentre]
          , [r].[SubJobQtyTotal]
          , [r].[IssMultLotsFlag]
          , [r].[ReservedLotSerFlag]
          , [r].[ReservedLotQty]
          , [r].[ReservedLot]
          , [r].[ReservedLotBin]
          , [r].[ReservedLotWarehouse]
          , [r].[ReservedLotQtyReserved]
          , [r].[ReservedLotQtyIssued]
          , [r].[AvailableLot]
          , [r].[AvailableLotBin]
          , [r].[AvailableLotWarehouse]
          , [r].[AvailableLotQtyOnHand]
          , [r].[AvailableLotExpiryDate]
          , [r].[AvailableLotCreationDate]
          , [r].[SubJobUom]
          , [RLK].[ReservedLots]
          , [r].[ReservedLotBleedNumber]
          , [r].[ReservedLotDonorNumber]
          , [r].[ReservedLotVendorBatchNumber]
          , [r].[ReservedLotOldLotNumber]
          , [r].[ReservedLotBleedDate]
    From    [Report].[Results_WipSubJobStock] As [r]
            Left Join [#ReservedLotKeys] As [RLK]
                On [RLK].[StartTime] = [r].[StartTime]
                   And [RLK].[StockCode] = [r].[StockCode]
    Where   [r].[StartTime] = @StartTime
    Order By [StockCode] Asc;



GO
EXEC sp_addextendedproperty N'MS_Description', N'iterates through all jobs and
subjobs, returning their stock - used in picklist/bill of materials reports',
'SCHEMA', N'Report', 'PROCEDURE', N'UspResults_WipSubJobStock', NULL, NULL
GO
```

Uses

[Process].[Status_WipSubJobStock]
[Report].[Results_WipSubJobStock]
[Process].[UspInsert_RedTagLogs]
[Report]

Author: Johnson, Chris

[Report].[UspResultsRefresh_TableTimes]

MS_Description

used to refresh all the lookup and history table, in addition providing times for how long this takes

Parameters

Name	Data Type	Max Length (Bytes)
@Exec	int	4
@HoursBetweenEachRun	numeric(5,2)	5

SQL Script

```
CREATE Proc [Report].[UspResultsRefresh_TableTimes]
    (
      @Exec Int
    , @HoursBetweenEachRun Numeric(5 , 2)
    )
As
    Begin

--remove nocount on to speed up query

        Set NoCount On;

        Declare @LookupStart DateTime2
          , @LookupEnd DateTime2
          , @HistoryStart DateTime2
          , @HistoryEnd DateTime2;

        Declare @GeneratedUsedByNameStart Varchar(500)= 'Development > Report System
Refresh > Exec: '
            + Convert(Varchar(5) , @Exec) + '; Hours between each run: '
            + Convert(Varchar(5) , @HoursBetweenEachRun) + '; Started: '
            + Convert(Varchar(24) , GetDate() , 113);

        Exec [Process].[UspInsert_RedTagLogs] @StoredProcDb = 'BlackBox' ,
            @StoredProcSchema = 'Report' ,
            @StoredProcName = 'UspResultsRefresh_TableTimes' ,
            @UsedByType = 'X' , @UsedByName = @GeneratedUsedByNameStart ,
            @UsedByDb = 'BlackBox';



        Create Table [#Results]
            (
              [SchemaName] Varchar(100)        collate latin1_general_bin
            , [TableName] Varchar(100)        collate latin1_general_bin
            , [LastUpdated] DateTime2
            , [OldRowCount] Int
            , [NewRowCount] Int
```

Author: Johnson, Chris

```
            , [UpdateStart] DateTime2
            , [UpdateEnd] DateTime2
            , [OldColumnCount] Int
            , [NewColumnCount] Int
            );

--Lookups

        Begin
            Select  @LookupStart = GetDate();

    --Get Preloaded figures

            Insert  [#Results]
                    ( [SchemaName]
                    , [TableName]
                    , [LastUpdated]
                    , [OldRowCount]
                    , [NewRowCount]
                    , [UpdateStart]
                    , [UpdateEnd]
                    , [OldColumnCount]
                    , [NewColumnCount]
                    )
                    Select  [s].[name]
                          , [t].[name]
                          , [LastUpdated] = [iu].[last_user_update]
                          , [OldRowCount] = [I].[row_count]
                          , [NewRowCount] = Null
                          , [UpdateStart] = @LookupStart
                          , [UpdateEnd] = Null
                          , [OldColumnCount] = Count([c].[name])
                          , [NewColumnCount] = Null
                    From    [sys].[tables] [t]
                            Inner Join [sys].[schemas] [s]
                                On [s].[schema_id] = [t].[schema_id]
                            Inner Join [sys].[dm_db_partition_stats] As [I]
                                On [t].[object_id] = [I].[object_id]
                                   And [I].[index_id] < 2
                            Left Join [sys].[columns] [c]
                                On [c].[object_id] = [t].[object_id]
                            Left Join [sys].[dm_db_index_usage_stats] [iu]
                                On [iu].[object_id] = [t].[object_id]
                    Where   [s].[name] = 'Lookups'
                    Group By [s].[name]
                          , [t].[name]
                          , [I].[row_count]
                          , [iu].[last_user_update];

            If @Exec = 1
                Begin
    --update tables

                    Exec [Process].[UspLoad_LoadController] @HoursBetweenEachRun =
@HoursBetweenEachRun;-- numeric

                End;
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults-Refresh_TableTimes

```
    --add results

            Update  [#Results]
            Set     [NewRowCount] = [t].[Row_Count]
                  , [NewColumnCount] = [t].[NewColumnCount]
            From    [#Results] [R]
                    Left Join ( Select  [SchemaName] = [s].[name]
                                      , [TableName] = [t].[name]
                                      , [Row_Count] = [I].[row_count]
                                      , [NewColumnCount] = Count([c].[name])
                                From    [sys].[tables] [t]
                                        Inner Join [sys].[schemas] [s]
                                            On [s].[schema_id] = [t].[schema_id]
                                        Inner Join [sys].[dm_db_partition_stats]
                                            As [I]
                                            On [t].[object_id] = [I].[object_id]
                                               And [I].[index_id] < 2
                                        Left Join [sys].[columns] [c]
                                            On [c].[object_id] = [t].[object_id]
                                Where   [s].[name] = 'Lookups'
                                Group By [s].[name]
                                      , [t].[name]
                                      , [I].[row_count]
                              ) [t]
                        On [t].[SchemaName] = [R].[SchemaName] Collate Latin1_-
General_BIN
                           And [t].[TableName] = [R].[TableName] Collate Latin1_-
General_BIN
            Where   [t].[Row_Count] Is Not Null;


            Select  @LookupEnd = GetDate();

            Update  [#Results]
            Set     [UpdateEnd] = @LookupEnd
            Where   [SchemaName] = 'Lookups';
        End;

--History

        Begin
            Select  @HistoryStart = GetDate();

    --Get Preloaded figures

            Insert  [#Results]
                    ( [SchemaName]
                    , [TableName]
                    , [LastUpdated]
                    , [OldRowCount]
                    , [NewRowCount]
                    , [UpdateStart]
                    , [UpdateEnd]
                    , [OldColumnCount]
                    , [NewColumnCount]
                    )
```

Author: Johnson, Chris

```
                    Select  [s].[name]
                          , [t].[name]
                          , [LastUpdated] = [iu].[last_user_update]
                          , [OldRowCount] = [I].[row_count]
                          , [NewRowCount] = Null
                          , [UpdateStart] = @LookupStart
                          , [UpdateEnd] = Null
                          , [OldColumnCount] = Count([c].[name])
                          , [NewColumnCount] = Null
                    From    [sys].[tables] [t]
                            Inner Join [sys].[schemas] [s]
                                On [s].[schema_id] = [t].[schema_id]
                            Inner Join [sys].[dm_db_partition_stats] As [I]
                                On [t].[object_id] = [I].[object_id]
                                   And [I].[index_id] < 2
                            Left Join [sys].[columns] [c]
                                On [c].[object_id] = [t].[object_id]
                            Left Join [sys].[dm_db_index_usage_stats] [iu]
                                On [iu].[object_id] = [t].[object_id]
                    Where   [s].[name] = 'History'
                    Group By [s].[name]
                          , [t].[name]
                          , [I].[row_count]
                          , [iu].[last_user_update];

            If @Exec = 1
                Begin
                    Exec [Process].[UspPopulate_HistoryTables1] @RebuildBit = 0; --
bit



                End;

            Update  [#Results]
            Set     [NewRowCount] = [t].[Row_Count]
                  , [NewColumnCount] = [t].[NewColumnCount]
            From    [#Results] [R]
                    Left Join ( Select  [SchemaName] = [s].[name]
                                      , [TableName] = [t].[name]
                                      , [Row_Count] = [I].[row_count]
                                      , [NewColumnCount] = Count(Distinct
[c].[name])
                                From    [sys].[tables] [t]
                                        Inner Join [sys].[schemas] [s]
                                            On [s].[schema_id] = [t].[schema_id]
                                        Inner Join [sys].[dm_db_partition_stats]
                                            As [I]
                                            On [t].[object_id] = [I].[object_id]
                                               And [I].[index_id] < 2
                                        Left Join [sys].[columns] [c]
                                            On [c].[object_id] = [t].[object_id]
                                Where   [s].[name] = 'History'
                                Group By [s].[name]
                                      , [t].[name]
                                      , [I].[row_count]
                              ) [t]
                        On [t].[SchemaName] = [R].[SchemaName] Collate Latin1_-
```

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults-Refresh_TableTimes

```
General_BIN
                            And [t].[TableName] = [R].[TableName] Collate Latin1_-
General_BIN
            Where   [t].[Row_Count] Is Not Null;

            Select  @HistoryEnd = GetDate();

            Update  [#Results]
            Set     [UpdateEnd] = @HistoryEnd
            Where   [SchemaName] = 'History';
        End;

        Declare @GeneratedUsedByNameEnd Varchar(500)= 'Development > Report System
Refresh > Exec: '
            + Convert(Varchar(5) , @Exec) + '; Hours between each run: '
            + Convert(Varchar(5) , @HoursBetweenEachRun) + '; Ended: '
            + Convert(Varchar(24) , GetDate() , 113);

        Exec [Process].[UspInsert_RedTagLogs] @StoredProcDb = 'BlackBox' ,
            @StoredProcSchema = 'Report' ,
            @StoredProcName = 'UspResultsRefresh_TableTimes' ,
            @UsedByType = 'X' , @UsedByName = @GeneratedUsedByNameEnd ,
            @UsedByDb = 'BlackBox';



        Set NoCount Off;

        Select  [SchemaName]
              , [TableName]
              , [LastUpdated]
              , [OldRowCount]
              , [NewRowCount]
              , [UpdateStart]
              , [UpdateEnd]
              , [OldColumnCount]
              , [NewColumnCount]
              , [SecondsToRun] = DateDiff(Second , [UpdateStart] , [UpdateEnd])
              , [MinutesToRun] = DateDiff(Minute , [UpdateStart] , [UpdateEnd])
              , [HoursToRun] = DateDiff(Hour , [UpdateStart] , [UpdateEnd])
        From    [#Results];

        --Drop Table [#Results];


    End;

GO
EXEC sp_addextendedproperty N'MS_Description', N'used to refresh all the lookup and
history table, in addition providing times for how long this takes', 'SCHEMA',
N'Report', 'PROCEDURE', N'UspResultsRefresh_TableTimes', NULL, NULL
GO
```

Uses

[Process].[UspInsert_RedTagLogs]

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Report.UspResults-Refresh_TableTimes

[Process].[UspLoad_LoadController]

[Process].[UspPopulate_HistoryTables1]

[Report]

Author: Johnson, Chris

[Reports].[UspResults_SearchForProcedures]

MS_Description

proc to search for stored procedures

Parameters

Name	Data Type	Max Length (Bytes)
@DbSearch	varchar(300)	300
@ProcedureSearch	varchar(150)	150
@SchemaSearch	varchar(150)	150
@ParameterSearch	varchar(150)	150

SQL Script

```
CREATE Proc [Reports].[UspResults_SearchForProcedures]
    (
      @DbSearch Varchar(300)
    , @ProcedureSearch Varchar(150)
    , @SchemaSearch Varchar(150)
    , @ParameterSearch Varchar(150)
    )
As /*
Stored Procedure created by Chris Johnson
2nd February 2016

The purpose of this stored procedure is to search for stored procs by name, schema,
parameters or database
*/
--Cater for nulls and append % that

    Set @ParameterSearch = '%' + Coalesce(@ParameterSearch , '') + '%';
    Set @ProcedureSearch = '%' + Coalesce(@ProcedureSearch , '') + '%';
    Set @SchemaSearch = '%' + Coalesce(@SchemaSearch , '') + '%';
    Set @DbSearch = '%' + Coalesce(@DbSearch , '') + '%';

    Declare @SQLScript Varchar(max);

    Create Table [#ResultSets]
        (
          [SchemaName] Varchar(300)            collate latin1_general_bin
        , [ProcedureName] Varchar(300)        collate latin1_general_bin
        , [ParameterDetails] Varchar(Max)    collate latin1_general_bin
        , [DatabaseName] Varchar(300)        collate latin1_general_bin
        , [ExecScript] Varchar(Max)            collate latin1_general_bin
--script to get distinct values of returned column

        , [CreateScript] Varchar(Max)        collate latin1_general_bin           -
-script to get top 10 with all fields from table

        );
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Reports.UspResults_-SearchForProcedures

```
    Set @SQLScript = 'Use [?];
    If Lower(Db_Name()) Like lower('''+@DbSearch+''')
    begin
    Insert  [#ResultSets]
        ( [SchemaName]
        , [ProcedureName]
        , [ParameterDetails]
        , [DatabaseName]
        , [ExecScript]
        , [CreateScript]
        )
        Select  [SchemaName] = [S].[name]
              , [ProcedureName] = [P].[name]
              , [ParameterDetails] = Stuff(( Select Distinct
                                                    '', ''
                                                    + QuoteName([P2].[name])
                                               From   [sys].[parameters] As [P2]
                                               Where  [P2].[object_id] =
[P].[object_id]
                                             For
                                               Xml Path('''')
                                             ) , 1 , 1 , '''')
              , [DatabaseName] = Db_Name()
              , [ExecScript] = ''exec '' + QuoteName([S].[name]) + ''.''
                + QuoteName([P].[name])
                + Coalesce(Stuff(( Select Distinct
                                          '', '' + [P2].[name]
                                          + '' = ''
                                   From     [sys].[parameters] As [P2]
                                   Where    [P2].[object_id] = [P].[object_id]
                                 For
                                   Xml Path('''')
                                 ) , 1 , 1 , '''') , '''')
              , [CreateScript] = Object_Definition([P].[object_id])
        From    [sys].[procedures] As [P]
                Left Join [sys].[schemas] As [S] On [S].[schema_id] =
[P].[schema_id]
        Where   Lower([S].[name]) Like Lower('''+@SchemaSearch+''')
                And Lower([P].[name]) Like Lower('''+@ProcedureSearch+''')
                And Lower(Stuff(( Select Distinct
                                          '', '' + QuoteName([P2].[name])
                                   From     [sys].[parameters] As [P2]
                                   Where    [P2].[object_id] = [P].[object_id]
                                 For
                                   Xml Path('''')
                                 ) , 1 , 1 , '''')) Like Lower('''+@Parameter-
Search+''');
end';


    Exec [Process].[ExecForEachDB] @cmd = @SQLScript;


    Select  [RS].[SchemaName]
          , [RS].[ProcedureName]
          , [RS].[ParameterDetails]
```

Author: Johnson, Chris

```
          , [RS].[DatabaseName]
          , [RS].[ExecScript]
          , [RS].[CreateScript]
    From    [#ResultSets] As [RS];
GO
EXEC sp_addextendedproperty N'MS_Description', N'proc to search for stored
procedures', 'SCHEMA', N'Reports', 'PROCEDURE', N'UspResults_SearchForProcedures',
NULL, NULL
GO
```

Uses

[Process].[ExecForEachDB]

[Reports]

Used By

[Report].[UspResults_RedTagDetails]

[Reports].[UspResults_SearchForText]

MS_Description

proc to search for text in dbs - warning this is server intensive

Parameters

Name	Data Type	Max Length (Bytes)
@TextToSearch	nvarchar(max)	max
@DBToSearch	nvarchar(500)	1000
@SchemaToSearch	nvarchar(500)	1000
@ObjectName	nvarchar(500)	1000
@TableView	varchar(10)	10
@ShowStats	bit	1
@IncludeExecErrors	bit	1

SQL Script

```
CREATE Proc [Reports].[UspResults_SearchForText]
    (
      @TextToSearch NVarchar(Max)
    , @DBToSearch NVarchar(500)
    , @SchemaToSearch NVarchar(500)
    , @ObjectName NVarchar(500)
    , @TableView Varchar(10)
    , @ShowStats Bit
    , @IncludeExecErrors Bit
    )
As
    Begin
        Set NoCount On;
        Declare @StartDate DateTime2 = GetDate();
        Declare @SQLScript NVarchar(Max);
        Set @TextToSearch = '%' + Coalesce(Lower(@TextToSearch) , '') + '%';
        Set @DBToSearch = '%' + Coalesce(Lower(@DBToSearch) , '') + '%';
        Set @SchemaToSearch = '%' + Coalesce(Lower(@SchemaToSearch) , '')
            + '%';
        Set @ObjectName = '%' + Coalesce(Lower(@ObjectName) , '') + '%';
        Set @TableView = Coalesce(Case When @TableView = '' Then Null
                                       Else Lower(@TableView)
                                  End , 'both');

        Create Table [#ObjectList]
            (
              [object_id] Int Not Null
            , [ObjectName] sysname                          collate latin1_general_bin
            , [ObjectType] Varchar(10)                      collate latin1_general_bin
            , [SchemaName] sysname                          collate latin1_general_bin
            , [DBName] Varchar(255)                           collate
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Reports.UspResults_-SearchForText

```
latin1_general_bin Not Null
            , [ObjectDefinition] NVarchar(Max)            collate latin1_general_bin
            , Constraint [PK_ObjList_ObjectID_DBName] Primary Key
                ( [object_id] , [DBName] )
            );

        Create Table [#ColumnList]
            (
              [object_id] Int Not Null
            , [ColumnID] Int Not Null
            , [ColumnName] sysname                        collate latin1_general_bin
            , [ColumnType] TinyInt
            , [max_length] Smallint
            , [DBName] Varchar(255)                        collate
latin1_general_bin Not Null
            , [ExecID] Int Identity(1 , 1)
            , Constraint [PL_ColList_ObjectID_ColID_DBName] Primary Key
                ( [object_id] , [ColumnID] , [DBName] )
            );

--Get list of table or views

        If @TableView In ( 'table' , 'both' )
            Begin
                Set @SQLScript = 'Use [?];
            Declare @DB Varchar(250)= Db_Name();
            if @DB not in(''tempdb'',''master'',''model'',''msdb'') and lower(@DB)
like '''
                    + @DBToSearch
                    + '''
            begin
                Insert [#ObjectList]
                        ( [object_id]
                        , [ObjectName]
                        , [ObjectType]
                        , [SchemaName]
                        , [DBName]
                        , [ObjectDefinition]
                        )
                Select  [T].[object_id]
                      , ObjectName = [T].[name]
                      , ObjectType = ''Table''
                      , SchemaName = [S].[name]
                      , DBName = @DB
                      , [ObjectDefinition]=null --Object_Definition([T].[object_id])
                From    [sys].[tables] As [T]
                        Left Join [sys].[schemas] As [S] On [S].[schema_id] =
[T].[schema_id]
                where Lower([T].[name]) Like ''' + @ObjectName + '''
            end';

                Exec [Process].[ExecForEachDB] @cmd = @SQLScript
            End;
        If @TableView In ( 'view' , 'both' )
            Begin
                Set @SQLScript = 'Use [?];
            Declare @DB Varchar(250)= Db_Name();
            if @DB not in(''tempdb'',''master'',''model'',''msdb'') and lower(@DB)
```

Author: Johnson, Chris

```
like '''
                    + @DBToSearch
                    + '''
            begin
                Insert [#ObjectList]
                        ( [object_id]
                        , [ObjectName]
                        , [ObjectType]
                        , [SchemaName]
                        , [DBName]
                        , [ObjectDefinition]
                        )
                Select  [V].[object_id]
                      , ObjectName = [V].[name]
                      , ObjectType = ''View''
                      , SchemaName = [S].[name]
                      , DBName = @DB
                      , Object_Definition([V].[object_id])
                From    [sys].[views]  As [V]
                        Left Join [sys].[schemas] As [S] On [S].[schema_id] =
[V].[schema_id]
                where Lower([V].[name]) Like ''' + @ObjectName + ''';
            End';

                Exec [Process].[ExecForEachDB] @cmd = @SQLScript;

            End;

--get list of columns that have a collation (so are text) and have a length that
will fit the search string

        Set @SQLScript = 'Use [?];
    Declare @DB Varchar(250)= Db_Name();
    if @DB not in(''tempdb'',''master'',''model'',''msdb'') and lower(@DB) like '''
            + @DBToSearch + '''
    begin
        Insert [#ColumnList]
                ( [object_id]
                , [ColumnID]
                , [ColumnName]
                , [ColumnType]
                , [max_length]
                , [DBName]
                )
        SELECT    [C].[object_id]
                , [C].[column_id]
                , [C].[name]
                , [C].[system_type_id]
                , [max_length]
                , @DB
         FROM sys.[columns] As [C]
         where collation_name is not null
         and [max_length]>=(' + Convert(Varchar(10) , Len(@TextToSearch))
            + '-2);
    end';

        Exec [sys].[sp_MSforeachdb] @SQLScript;
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Reports.UspResults_-SearchForText

```
        Declare @CurrentID Int
          , @MaxID Int
          , @CurrentCount Int;

        Create NonClustered Index [ColList_ExID_ColName_ObjID_DB] On [#ColumnList]
([ExecID],[ColumnName],[object_id],[DBName]);
        Create NonClustered Index [ObjList_ObjID_ObName_Schema_DB] On [#ObjectList]
([object_id],[ObjectName],[SchemaName],[DBName]);

        Create Table [#Results]
            (
              [ColumnName] sysname                collate latin1_general_bin
            , [TableName] sysname                collate latin1_general_bin
            , [SchemaName] sysname                collate latin1_general_bin
            , [DBName] Varchar(250)                collate latin1_general_bin
            , [CountofMatchingRows] Int
            , [MaxMatchingRow] NVarchar(Max)    collate latin1_general_bin
            , [MinMatchingRow] NVarchar(Max)    collate latin1_general_bin
            );

        Select  @CurrentID = Min([CL].[ExecID])
        From    [#ColumnList] As [CL];

        Select  @MaxID = Max([CL].[ExecID])
        From    [#ColumnList] As [CL];

        If @ShowStats = 1
            Begin
                Print 'Columns to check ' + Convert(NVarchar(50) , @MaxID);
            End;

        Declare @CurrentIDtxt NVarchar(50)
          , @CurrentPerct Numeric(20 , 2)
          , @CurrentPerctTxt NVarchar(50)
          , @TimeSpent Numeric(20 , 2)
          , @TimeSpentTxt NVarchar(50)
          , @TimeRemaining Numeric(20 , 2)
          , @TimeRemainingTxt NVarchar(50)
          , @TimeOffset Numeric(20 , 2);

        --time offset removes initial load time

        Select  @TimeOffset = DateDiff(Second , @StartDate , GetDate());

        While @CurrentID <= @MaxID
            Begin
                Select  @SQLScript = 'Insert [#Results]
                        ( [ColumnName]
                        , [TableName]
                        , [SchemaName]
                        , [DBName]
                        , [CountofMatchingRows]
                        , [MaxMatchingRow]
                        , [MinMatchingRow]
                        )
                        Select [ColumnName]=''' + [CL].[ColumnName] + '''
```

Author: Johnson, Chris

```
                        , [TableName]=''' + [OL].[ObjectName] + '''
                        , [SchemaName]=''' + [OL].[SchemaName] + '''
                        , [DBName]=''' + [CL].[DBName] + '''
                        , [CountofMatchingRows] = count(1)
                        , [MaxMatchingRow] = convert(Nvarchar(max),max('
                        + QuoteName([CL].[ColumnName]) + '))
                        , [MinMatchingRow] = convert(Nvarchar(max),min('
                        + QuoteName([CL].[ColumnName]) + '))
                        from ' + QuoteName([CL].[DBName]) + '.'
                        + QuoteName([OL].[SchemaName]) + '.'
                        + QuoteName([OL].[ObjectName]) + '
                        where lower(' + QuoteName([CL].[ColumnName])
                        + ') collate Latin1_General_CI_AI like '''
                        + @TextToSearch + ''' collate Latin1_General_CI_AI
                        having count(1)>0'
                From    [#ColumnList] As [CL]
                        Left Join [#ObjectList] As [OL]
                            On [OL].[object_id] = [CL].[object_id]
                               And [OL].[DBName] = [CL].[DBName]
                Where   [CL].[ExecID] = @CurrentID;

                Begin Try
                    Print @SQLScript;
                    Exec (@SQLScript);
                End Try
                Begin Catch
                    If @IncludeExecErrors = 1
                        Begin
                            Insert  [#Results]
                                    ( [ColumnName]
                                    , [TableName]
                                    , [SchemaName]
                                    , [DBName]
                                    , [CountofMatchingRows]
                                    , [MaxMatchingRow]
                                    , [MinMatchingRow]
                                    )
                                    Select  [CL].[ColumnName]
                                          , [OL].[ObjectName]
                                          , [OL].[SchemaName]
                                          , [CL].[DBName]
                                          , 0
                                          , 'Unable to execute against'
                                          , 'Unable to execute against'
                                    From    [#ColumnList] As [CL]
                                            Left Join [#ObjectList] As [OL]
                                                On [OL].[object_id] =
[CL].[object_id]
                                                   And [OL].[DBName] = [CL].[DBName]
                                    Where   [CL].[ExecID] = @CurrentID;
                        End;
                End Catch;
                If @ShowStats = 1
                    Begin
                        If @CurrentID % 1000 = 0
                            Begin
                                Set @CurrentIDtxt = Convert(NVarchar(50) , @Current-
```

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Reports.UspResults_-SearchForText

```
ID);
                                Set @CurrentPerct = @CurrentID
                                    / Convert(Numeric(20 , 2) , @MaxID) * 100;
                                Set @CurrentPerctTxt = Convert(NVarchar(50) ,
@CurrentPerct);
                                Set @TimeSpent = DateDiff(Second , @StartDate ,
                                                          GetDate())
                                    - @TimeOffset;
                                Set @TimeSpentTxt = Convert(NVarchar(50) , @Time-
Spent);
                                Set @TimeRemaining = ( @TimeSpent
                                                       / @CurrentPerct * 100 )
                                    - @TimeSpent;
                                Set @TimeRemainingTxt = Convert(NVarchar(50) , @Time-
Remaining);

                                Print @CurrentIDtxt + ' Columns checked - '
                                    + @CurrentPerctTxt + '% - Time spent '
                                    + @TimeSpentTxt
                                    + ' Seconds - estimated remaining Time '
                                    + @TimeRemainingTxt + ' Seconds';
                            End;
                    End;
                Delete  [#ColumnList]
                Where   [ExecID] = @CurrentID;

                Select  @CurrentID = Min([CL].[ExecID])
                From    [#ColumnList] As [CL];
            End;

        Set NoCount Off;
        Select  [DBName] = QuoteName([R].[DBName])
              , [Dot1] = '.'
              , [SchemaName] = QuoteName([R].[SchemaName])
              , [Dot2] = '.'
              , [TableName] = QuoteName([R].[TableName])
              , [ColumnName] = QuoteName([R].[ColumnName])
              , [Script] = 'SELECT * FROM ' + QuoteName([R].[DBName]) + '.'
                + QuoteName([R].[SchemaName]) + '.'
                + QuoteName([R].[TableName]) + '
where lower(' + QuoteName([R].[ColumnName]) + ') like ''' + @TextToSearch
                + ''''
              , [R].[CountofMatchingRows]
              , [R].[MaxMatchingRow]
              , [R].[MinMatchingRow]
        From    [#Results] As [R]
        Order By [R].[CountofMatchingRows] Desc;

        Drop Table [#ObjectList];
        Drop Table [#ColumnList];
        Drop Table [#Results];
    End;
GO
EXEC sp_addextendedproperty N'MS_Description', N'proc to search for text in dbs -
warning this is server intensive', 'SCHEMA', N'Reports', 'PROCEDURE', N'UspResults_-
SearchForText', NULL, NULL
GO
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Reports.UspResults_-SearchForText

Uses

[Process].[ExecForEachDB]
[Reports]

Author: Johnson, Chris

[Review].[UspResults_CompanyDuplicates]

MS_Description

check for duplicates in companyname table

Parameters

Name	Data Type	Max Length (Bytes)
@IncludeNotes	bit	1
@RedTagType	char	1
@RedTagUse	varchar(500)	500

SQL Script

```
CREATE Proc [Review].[UspResults_CompanyDuplicates]
    (
      @IncludeNotes Bit
    , @RedTagType Char(1)
    , @RedTagUse Varchar(500)
    )
As
    Begin
    Set NoCount On
    Select  @IncludeNotes = Coalesce(@IncludeNotes , 0);

    --Red tag

        Declare @RedTagDB Varchar(255)= Db_Name();
        Exec [Process].[UspInsert_RedTagLogs] @StoredProcDb = 'BlackBox' ,
            @StoredProcSchema = 'Review' ,
            @StoredProcName = 'UspResults_CompanyDuplicates' ,
            @UsedByType = @RedTagType , @UsedByName = @RedTagUse ,
            @UsedByDb = @RedTagDB;

        Create Table [#CpReview]
            (
              [Company] Varchar(150)        collate latin1_general_bin
            , [CompanyName] Varchar(250)    collate latin1_general_bin
            , [ShortName] Varchar(250)        collate latin1_general_bin
            , [RecordCount] Int
            , RowType Varchar(50)            collate latin1_general_bin
            );

        If @IncludeNotes=1
        BEGIN
            Insert [#CpReview]
                    ( [Company]
                    , [CompanyName]
                    , [ShortName]
                    , [RecordCount]
```

Author: Johnson, Chris

```
                    , [RowType]
                    )
            Values  ( 'List of '  -- Company - varchar(150)

                    , 'companies that have been'  -- CompanyName - varchar(250)

                    , 'duplicated in lookup table'  -- ShortName - varchar(250)

                    , 2  -- RecordCount - int

                    , 'Notes'  -- RowType - varchar(50)

                    )
        END

        Insert [#CpReview]
                ( [Company]
                , [CompanyName]
                , [ShortName]
                , [RecordCount]
                , [RowType]
                )
        Select  [CN].[Company]
              , [CN].[CompanyName]
              , [CN].[ShortName]
              , [RecordCount] = Count(1)
              , RowType ='Data'
        From    [Lookups].[CompanyNames] [CN]
        Group By [CN].[Company]
              , [CN].[CompanyName]
              , [CN].[ShortName]
        Having  Count(1) > 1;

        SELECT [CR].[Company]
             , [CR].[CompanyName]
             , [CR].[ShortName]
             , [CR].[RecordCount]
             , [CR].[RowType]
        From [#CpReview] [CR]
        Order By [CR].[RowType] Desc
    End;
GO
EXEC sp_addextendedproperty N'MS_Description', N'check for duplicates in companyname
table', 'SCHEMA', N'Review', 'PROCEDURE', N'UspResults_CompanyDuplicates', NULL,
NULL
GO
```

Uses

[Lookups].[CompanyNames]
[Process].[UspInsert_RedTagLogs]
[Review]

Author: Johnson, Chris

[Review].[UspResults_CompanyNamesMissing]

MS_Description

check for missing company names in lookup

Parameters

Name	Data Type	Max Length (Bytes)
@RedTagType	char	1
@RedTagUse	varchar(500)	500

SQL Script

```
CREATE Proc [Review].[UspResults_CompanyNamesMissing]
    (
      @RedTagType Char(1)
    , @RedTagUse Varchar(500)
    )
As
    Begin
        Set NoCount On;

    --Red tag

        Declare @RedTagDB Varchar(255)= Db_Name();
        Exec [Process].[UspInsert_RedTagLogs] @StoredProcDb = 'BlackBox' ,
            @StoredProcSchema = 'Review' ,
            @StoredProcName = 'UspResults_CompanyNamesMissing' ,
            @UsedByType = @RedTagType , @UsedByName = @RedTagUse ,
            @UsedByDb = @RedTagDB;


        Create Table [#ListOfDbs]
            (
              [DatabaseName] sysname Collate Latin1_General_BIN
            );

        Declare @SQLCmd Varchar(Max)= N'use [?];
If Lower(db_name()) Like ''sysprocompany%'' And Lower(db_name()) Not Like ''%_srs''
begin
Insert [#ListOfDbs]
        ( [DatabaseName] )
Select replace(lower(db_name()),''sysprocompany'','''')
end';

        Exec [Process].[ExecForEachDB] @cmd = @SQLCmd;

        Set NoCount Off;
        Select  'SysproCompany' + [LOD].[DatabaseName] As [DatabaseWithoutCompany-
Name]
              , [CN].[CompanyName]
```

Author: Johnson, Chris

```
                , [CN].[ShortName]
                , [CN].[Currency]
        From    [#ListOfDbs] [LOD]
                Left Join [Lookups].[CompanyNames] [CN]
                    On [CN].[Company] = [LOD].[DatabaseName]
        Where   [CN].[Company] Is Null
                And IsNumeric([LOD].[DatabaseName]) = 1;

        Set NoCount On;
        Drop Table [#ListOfDbs];
    End;
GO
EXEC sp_addextendedproperty N'MS_Description', N'check for missing company names in lookup', 'SCHEMA', N'Review', 'PROCEDURE', N'UspResults_CompanyNamesMissing', NULL, NULL
GO
```

Uses

[Lookups].[CompanyNames]
[Process].[ExecForEachDB]
[Process].[UspInsert_RedTagLogs]
[Review]

Author: Johnson, Chris

[Review].[UspResults_CurrencyRatesMissing]

MS_Description

check that currency rates are available

Parameters

Name	Data Type	Max Length (Bytes)
@RedTagType	char	1
@RedTagUse	varchar(500)	500

SQL Script

```
CREATE Proc [Review].[UspResults_CurrencyRatesMissing]
    (
      @RedTagType Char(1)
    , @RedTagUse Varchar(500)
    )
As
    Begin
        Set NoCount On;

    --Red tag

        Declare @RedTagDB Varchar(255)= Db_Name();
        Exec [Process].[UspInsert_RedTagLogs] @StoredProcDb = 'BlackBox' ,
            @StoredProcSchema = 'Review' ,
            @StoredProcName = 'UspResults_CurrencyRatesMissing' ,
            @UsedByType = @RedTagType , @UsedByName = @RedTagUse ,
            @UsedByDb = @RedTagDB;

        Set NoCount Off;
        If Not Exists ( Select  1
                        From    [Lookups].[CurrencyRates] [CR]
                        Where   GetDate() Between [CR].[StartDateTime]
                                          And     [CR].[EndDateTime] )
            Begin
                Select  [CurrencyStatus] = 'No current exchange rates';
            End;
        If Exists ( Select  1
                    From    [Lookups].[CurrencyRates] [CR]
                    Where   GetDate() Between [CR].[StartDateTime]
                                      And     [CR].[EndDateTime] )
            Begin
                Select  [CurrencyStatus] = 'Current exchange rates';
            End;
    End;
GO
EXEC sp_addextendedproperty N'MS_Description', N'check that currency rates are
available', 'SCHEMA', N'Review', 'PROCEDURE', N'UspResults_CurrencyRatesMissing',
NULL, NULL
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Review.UspResults_-CurrencyRatesMissing

```
GO
```

Uses

[Lookups].[CurrencyRates]

[Process].[UspInsert_RedTagLogs]

[Review]

Author: Johnson, Chris

[Review].[UspResults_DbDiffs]

MS_Description

provide details of differences between databases

Parameters

Name	Data Type	Max Length (Bytes)
@RedTagType	char	1
@RedTagUse	varchar(500)	500

SQL Script

```
CREATE Proc [Review].[UspResults_DbDiffs]
    (
      @RedTagType Char(1)
    , @RedTagUse Varchar(500)
    )
As
    Begin

        Set NoCount On;
    --Red tag

        Declare @RedTagDB Varchar(255)= Db_Name();
        Exec [Process].[UspInsert_RedTagLogs] @StoredProcDb = 'BlackBox' ,
            @StoredProcSchema = 'Review' ,
            @StoredProcName = 'UspResults_DbDiffs' , @UsedByType = @RedTagType ,
            @UsedByName = @RedTagUse , @UsedByDb = @RedTagDB;


        Create Table [#ObjectsAndColumns]
            (
              [SchemaName] sysname        collate latin1_general_bin
            , [ObjectName] sysname        collate latin1_general_bin
            , [ColumnName] sysname        collate latin1_general_bin Null
            , [DatabaseName] sysname    collate latin1_general_bin
            , [ColumnType] sysname        collate latin1_general_bin Null
            , [max_length] Smallint
            , [precision] TinyInt
            , [ObjectType] Varchar(50)    collate latin1_general_bin
            , [parameters] sysname        collate latin1_general_bin Null
            );

        Declare @DBCount Int;
        Declare @SQLcmd NVarchar(Max) = 'Use [?];
If Lower(Db_Name()) Not Like ''%_srs'' And Lower(Db_Name()) Like ''sysprocompany%''
    Begin
        Insert [#ObjectsAndColumns]
        ( [SchemaName]
        , [ObjectName]
```

Author: Johnson, Chris

```
        , [ColumnName]
        , [DatabaseName]
        , [ColumnType]
        , [max_length]
        , [precision]
        , [ObjectType]
        )
    Select  [SchemaName] = [S].[name]
            , [TableName] = [T].[name]
            , [ColumnName] = [C].[name]
            , [DatabaseName] = Db_Name()
            , [ColumnType] = [Typ].[name]
            , [C].[max_length]
            , [C].[precision]
            , [ObjectType] = ''Table''
    From    [sys].[tables] [T]
            left Join [sys].[columns] [C]
                On [C].[object_id] = [T].[object_id]
            Inner Join [sys].[schemas] [S]
                On [S].[schema_id] = [T].[schema_id]
            Left Join [sys].[types] [Typ]
                On [Typ].[system_type_id] = [C].[system_type_id];
    End;';
        Declare @SQLcmdview NVarchar(Max) = 'Use [?];
If Lower(Db_Name()) Not Like ''%_srs'' And Lower(Db_Name()) Like ''sysprocompany%''
    Begin
        Insert [#ObjectsAndColumns]
        ( [SchemaName]
        , [ObjectName]
        , [ColumnName]
        , [DatabaseName]
        , [ColumnType]
        , [max_length]
        , [precision]
        , [ObjectType]
        )
    Select  [SchemaName] = [S].[name]
            , [TableName] = [T].[name]
            , [ColumnName] = [C].[name]
            , [DatabaseName] = Db_Name()
            , [ColumnType] = [Typ].[name]
            , [C].[max_length]
            , [C].[precision]
            , [ObjectType] = ''View''
    From    [sys].[views] [T]
            left Join [sys].[columns] [C]
                On [C].[object_id] = [T].[object_id]
            Inner Join [sys].[schemas] [S]
                On [S].[schema_id] = [T].[schema_id]
            Left Join [sys].[types] [Typ]
                On [Typ].[system_type_id] = [C].[system_type_id];
    End;';
        Declare @SQLcmdProc NVarchar(Max) = 'Use [?];
If Lower(Db_Name()) Not Like ''%_srs'' And Lower(Db_Name()) Like ''sysprocompany%''
    Begin
        Insert  [#ObjectsAndColumns]
        ( [SchemaName]
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Review.UspResults_Db-Diffs

```
        , [ObjectName]
        , [DatabaseName]
        , [ObjectType]
        , [parameters]
        )
        Select  [S].[name]
              , [P].[name]
              , Db_Name()
              , ''StoredProc''
              , PT.name
        From    [sys].[schemas] [S]
                Inner Join [sys].[procedures] [P]
                    On [P].[schema_id] = [S].[schema_id]
                left join sys.parameters [PT]
                on [PT].[object_id] = [P].[object_id]
    End;';




        Exec [Process].[ExecForEachDB] @cmd = @SQLcmd;
        Exec [Process].[ExecForEachDB] @cmd = @SQLcmdview;
        Exec [Process].[ExecForEachDB] @cmd = @SQLcmdProc;

        Select  @DBCount = Count(Distinct [TAC].[DatabaseName])
        From    [#ObjectsAndColumns] [TAC];

        Set NoCount Off;
        Select  [TAC].[SchemaName]
              , [TAC].[ObjectName]
              , [TAC].[ObjectType]
              , [TAC].[ColumnName]
              , [Parameters] = [TAC].[parameters]
              , [TAC].[ColumnType]
              , [Max_Length] = [TAC].[max_length]
              , [Precision] = [TAC].[precision]
              , [DbWhereExists] = Count(Distinct [TAC].[DatabaseName])
              , [TotalDbCount] = @DBCount
              , [MinDatabaseName] = Min([TAC].[DatabaseName])
              , [MaxDatabaseName] = Max([TAC].[DatabaseName])
        From    [#ObjectsAndColumns] [TAC]
        Group By [TAC].[SchemaName]
              , [TAC].[ObjectName]
              , [TAC].[ObjectType]
              , [TAC].[ColumnName]
              , [TAC].[ColumnType]
              , [TAC].[max_length]
              , [TAC].[precision]
              , [TAC].[parameters]
        Having  Count([TAC].[DatabaseName]) <> @DBCount
        Order By [TAC].[ObjectType] Desc;


        Drop Table [#ObjectsAndColumns];
    End;
GO
EXEC sp_addextendedproperty N'MS_Description', N'provide details of differences
```

Author: Johnson, Chris

```
between databases', 'SCHEMA', N'Review', 'PROCEDURE', N'UspResults_DbDiffs', NULL,
NULL
GO
```

Uses

[Process].[ExecForEachDB]

[Process].[UspInsert_RedTagLogs]

[Review]

Author: Johnson, Chris

[Review].[UspResults_LastUpdatedTimes]

MS_Description

check lookup and history tables for when last updated

Parameters

Name	Data Type	Max Length (Bytes)
@MinutesToCheck	int	4
@RedTagType	char	1
@RedTagUse	varchar(500)	500

SQL Script

```
CREATE Proc [Review].[UspResults_LastUpdatedTimes]
    (
      @MinutesToCheck int
    , @RedTagType Char(1)
    , @RedTagUse Varchar(500)
    )
As
    Begin
        Set NoCount On;
        Select  @MinutesToCheck = Coalesce(@MinutesToCheck , 30);

    --Red tag

        Declare @RedTagDB Varchar(255)= Db_Name();
        Exec [Process].[UspInsert_RedTagLogs] @StoredProcDb = 'BlackBox' ,
            @StoredProcSchema = 'Review' ,
            @StoredProcName = 'UspResults_LastUpdatedTimes' ,
            @UsedByType = @RedTagType , @UsedByName = @RedTagUse ,
            @UsedByDb = @RedTagDB;


        Declare @TableWithSchema Varchar(500)
          , @SqlScript NVarchar(Max)
          , @cmd NVarchar(Max);

        Create Table [#TablesNotUpdated]
            (
              [TableName] Varchar(500)    collate Latin1_General_BIN
            , [LastUpdated] DateTime2
            );

        Declare [Tables] Cursor
        For
            Select  Distinct
                    [Tables] = QuoteName([S].[name]) + '.'
                    + QuoteName([T].[name])--,[C].[name]
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Review.UspResults_Last-UpdatedTimes

```
            From    [sys].[columns] [C]
                    Inner Join [sys].[tables] [T]
                        On [T].[object_id] = [C].[object_id]
                    Inner Join [sys].[schemas] [S]
                        On [S].[schema_id] = [T].[schema_id]
            Where   Lower([C].[name]) = 'lastupdated'
            Order By [Tables] Asc;

        Open [Tables];

        Fetch Next From [Tables] Into @TableWithSchema;
 --Get first database to execute against


        While @@fetch_status = 0 --when fetch is successful

            Begin
                Set @cmd = 'Insert [#TablesNotUpdated]
                                ( [TableName] , [LastUpdated] )
                                SELECT ''' + @TableWithSchema
                    + ''',Max([LastUpdated]) FROM ' + @TableWithSchema;
                Exec ( @cmd );


                Fetch Next From [Tables] Into @TableWithSchema;--Get next database
to execute against

            End;

        Close [Tables];
        Deallocate [Tables];

        Set NoCount Off;
        Select  [TNU].[TableName]
              , [TNU].[LastUpdated]
              , [MinutesSinceRun] = DateDiff(Minute , [TNU].[LastUpdated] ,
                                             GetDate())
                                             ,Test = @MinutesToCheck
        From    [#TablesNotUpdated] [TNU]
        Where   DateDiff(Minute , [TNU].[LastUpdated] , GetDate()) > @MinutesTo-
Check;
        Set NoCount On;
        Drop Table [#TablesNotUpdated];
    End;
GO
EXEC sp_addextendedproperty N'MS_Description', N'check lookup and history tables for
when last updated', 'SCHEMA', N'Review', 'PROCEDURE', N'UspResults_LastUpdated-
Times', NULL, NULL
GO
```

Uses

[Process].[UspInsert_RedTagLogs]
[Review]

Author: Johnson, Chris

[Review].[UspResults_LogStats]

MS_Description

review of red tag logs

Parameters

Name	Data Type	Max Length (Bytes)
@RedTagType	char	1
@RedTagUse	varchar(500)	500

SQL Script

```
CREATE Proc [Review].[UspResults_LogStats]
    (
      @RedTagType Char(1)
    , @RedTagUse Varchar(500)
    )
As
    Begin
        Set NoCount On;

    --Red tag

        Declare @RedTagDB Varchar(255)= Db_Name();
        Exec [Process].[UspInsert_RedTagLogs] @StoredProcDb = 'BlackBox' ,
            @StoredProcSchema = 'Review' ,
            @StoredProcName = 'UspResults_LogStats' ,
            @UsedByType = @RedTagType , @UsedByName = @RedTagUse ,
            @UsedByDb = @RedTagDB;
        Select  [RTL].[TagID]
              , [RTL].[StoredProcDb]
              , [RTL].[StoredProcSchema]
              , [RTL].[StoredProcName]
              , [UsedByType] = [RTUBT].[UsedByDescription]
              , [RTL].[UsedByName]
              , [RTL].[UsedByDb]
              , [ReportHour] = DateName(Hour , [RTL].[TagDatetime])
              , [ReportDay] = DateName(Weekday , [RTL].[TagDatetime])
              , [TagDatetime] = Convert(DateTime , ( Left(Convert(Varchar(255) ,
[RTL].[TagDatetime]) ,
                                                         18) + '0' ))
              , [DaysSinceReportRun] = DateDiff(Day , [RTL].[TagDatetime] ,
                                                GetDate())
        From    [BlackBox].[History].[RedTagLogs] [RTL]
                Left Join [Lookups].[RedTagsUsedByType] [RTUBT]
                    On [RTUBT].[UsedByType] = [RTL].[UsedByType]
        Order By [RTL].[TagID] Desc;
    End;
GO
EXEC sp_addextendedproperty N'MS_Description', N'review of red tag logs', 'SCHEMA',
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Review.UspResults_Log-Stats

```
N'Review', 'PROCEDURE', N'UspResults_LogStats', NULL, NULL
GO
```

Uses

[History].[RedTagLogs]
[Lookups].[RedTagsUsedByType]
[Process].[UspInsert_RedTagLogs]
[Review]

Author: Johnson, Chris

[Review].[UspResults_LookupDuplicates]

MS_Description

check lookup tables for duplicates (indicates that refresh failed)

Parameters

Name	Data Type	Max Length (Bytes)
@RedTagType	char	1
@RedTagUse	varchar(500)	500

SQL Script

```
CREATE Proc [Review].[UspResults_LookupDuplicates]
    (
      @RedTagType Char(1)
    , @RedTagUse Varchar(500)
    )
As
    Begin
        Set NoCount On;

    --Red tag

        Declare @RedTagDB Varchar(255)= Db_Name();
        Exec [Process].[UspInsert_RedTagLogs] @StoredProcDb = 'BlackBox' ,
            @StoredProcSchema = 'Review' ,
            @StoredProcName = 'UspResults_LookupDuplicates' ,
            @UsedByType = @RedTagType , @UsedByName = @RedTagUse ,
            @UsedByDb = @RedTagDB;

        Create Table [#DupeCheck]
            (
              [TableName] Varchar(150) Collate Latin1_General_BIN
            , [Company] Varchar(150) Collate Latin1_General_BIN
            , [ID] Varchar(500) Collate Latin1_General_BIN
            , [DescriptionField] Varchar(1000) Collate Latin1_General_BIN
            , [DupeCount] Int
            );

        Insert  [#DupeCheck]
                ( [TableName]
                , [Company]
                , [ID]
                , [DescriptionField]
                , [DupeCount]
                )
                Select  [t].[TableName]
                      , [t].[Company]
                      , [t].[ID]
                      , [t].[DescriptionField]
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Review.UspResults_-LookupDuplicates

```
                        , [t].[EntryCount]
                From    ( Select    [TableName] = '[Lookups].[AdmTransactionIDs]'
                                  , [Company] = 'All'
                                  , [ID] = Convert(Varchar(150) ,
[ATID].[TransactionId])
                                  , [DescriptionField] = Convert(Varchar(500) ,
[ATID].[TransactionDescription])
                                  , [EntryCount] = Count(1)
                          From      [Lookups].[AdmTransactionIDs] [ATID]
                          Group By  Convert(Varchar(150) , [ATID].[TransactionId])
                                  , [ATID].[TransactionDescription]
                          Having    Count(1) > 1
                          Union
                          Select    [TableName] = '[Lookups].[ApJnlDistribExpense-
Type]'
                                  , [Company] = 'All'
                                  , [ID] = Convert(Varchar(150) , [AJDET].[Expense-
Type])
                                  , [DescriptionField] = Convert(Varchar(500) ,
[AJDET].[ExpenseTypeDesc])
                                  , [EntryCount] = Count(1)
                          From      [Lookups].[ApJnlDistribExpenseType] [AJDET]
                          Group By  Convert(Varchar(150) , [AJDET].[ExpenseType])
                                  , [AJDET].[ExpenseTypeDesc]
                          Having    Count(1) > 1
                          Union
                          Select    [TableName] = '[Lookups].[ArInvoicePayTrnType]'
                                  , [Company] = 'All'
                                  , [ID] = Convert(Varchar(150) , [AIPTT].[TrnType])
                                  , [DescriptionField] = Convert(Varchar(500) ,
[AIPTT].[TrnTypeDesc])
                                  , Count(1)
                          From      [Lookups].[ArInvoicePayTrnType] [AIPTT]
                          Group By  Convert(Varchar(150) , [AIPTT].[TrnType])
                                  , [AIPTT].[TrnTypeDesc]
                          Having    Count(1) > 1
                          Union
                          Select    [TableName] = '[Lookups].[BankBalances]'
                                  , [Company] = [BB].[DatabaseName]
                                  , [ID] = Convert(Varchar(150) , [BB].[Bank])
                                    + Convert(Varchar(150) , [BB].[DateTimeOf-
Balance])
                                  , [DescriptionField] = Convert(Varchar(500) ,
[BB].[BankDescription])
                                  , Count(1)
                          From      [Lookups].[BankBalances] [BB]
                          Group By  Convert(Varchar(150) , [BB].[Bank])
                                    + Convert(Varchar(150) , [BB].[DateTimeOf-
Balance])
                                  , [BB].[DatabaseName]
                                  , Convert(Varchar(500) , [BB].[BankDescription])
                          Having    Count(1) > 1
                          Union
                          Select    [TableName] = '[Lookups].[Bin]'
                                  , [Company] = [B].[Company]
                                  , [B].[Bin]
                                  , [DescriptionField] = Convert(Varchar(500) ,
Null)
                                  , [EntryCount] = Count(1)
```

Author: Johnson, Chris

```
                            From      [Lookups].[Bin] [B]
                            Group By  [B].[Company]
                                    , [B].[Bin]
                            Having    Count(1) > 1
                            Union
                            Select    [TableName] = '[Lookups].[BudgetType]'
                                    , [Company] = 'All'
                                    , [ID] = Convert(Varchar(150) , [BT].[BudgetType])
                                    , [DescriptionField] = Convert(Varchar(500) ,
[BT].[BudgetTypeDesc])
                                    , [EntryCount] = Count(1)
                            From      [Lookups].[BudgetType] [BT]
                            Group By  [BT].[BudgetType]
                                    , [BT].[BudgetTypeDesc]
                            Having    Count(1) > 1
                            Union
                            Select    [TableName] = '[Lookups].[CompanyNames]'
                                    , [Company] = [CN].[Company]
                                    , [ID] = [CN].[Company]
                                    , [DescriptionField] = Convert(Varchar(500) , (
[CN].[ShortName]
                                                                    + ' - '
                                                                    + [CN].[CompanyName]
                                                                    + ' - '
                                                                    + [CN].[Currency] ))
                                    , [EntryCount] = Count(1)
                            From      [Lookups].[CompanyNames] [CN]
                            Group By  [CN].[ShortName] + ' - '
                                      + [CN].[CompanyName] + ' - '
                                      + [CN].[Currency]
                                    , [CN].[Company]
                                    , [CN].[Company]
                            Having    Count(1) > 1
                            Union
                            Select    [TableName] = '[Lookups].[CurrencyRates]'
                                    , [Company] = 'ALL'
                                    , [ID] = Convert(Varchar(150) , [CR].[StartDate-
Time])
                                      + ' - '
                                      + Convert(Varchar(150) , [CR].[EndDateTime])
                                      + ' ' + [CR].[Currency]
                                    , [DescriptionField] = Convert(Varchar(500) ,
Null)
                                    , [EntryCount] = Count(1)
                            From      [Lookups].[CurrencyRates] [CR]
                            Group By  Convert(Varchar(150) , [CR].[StartDateTime])
                                      + ' - '
                                      + Convert(Varchar(150) , [CR].[EndDateTime])
                                      + ' ' + [CR].[Currency]
                            Having    Count(1) > 1
                            Union
                            Select    [TableName] = '[Lookups].[GenJournalCtlJnl-
Source]'
                                    , [Company] = 'ALL'
                                    , [ID] = Convert(Varchar(150) , [GJCJS].[Gen-
JournalCtlJnlSource])
                                    , [DescriptionField] = Convert(Varchar(500) ,
[GJCJS].[GenJournalCtlJnlSourceDesc])
                                    , [EntryCount] = Count(1)
```

Author: Johnson, Chris

```
                            From      [Lookups].[GenJournalCtlJnlSource] [GJCJS]
                            Group By  [GJCJS].[GenJournalCtlJnlSource]
                                    , [GJCJS].[GenJournalCtlJnlSourceDesc]
                            Having    Count(1) > 1
                            Union
                            Select    [TableName] = '[Lookups].[GenJournalCtlSource]'
                                    , [Company] = 'ALL'
                                    , [ID] = Convert(Varchar(150) , [GJCS].[Source])
                                    , [DescriptionField] = Convert(Varchar(500) ,
[GJCS].[SourceDescription])
                                    , [EntryCount] = Count(1)
                            From      [Lookups].[GenJournalCtlSource] [GJCS]
                            Group By  [GJCS].[Source]
                                    , [GJCS].[SourceDescription]
                            Having    Count(1) > 1
                            Union
                            Select    [TableName] = '[Lookups].[GenJournalDetail-
Source]'
                                    , [Company] = 'ALL'
                                    , [ID] = Convert(Varchar(150) , [GJDS].[GJSource])
                                    , [DescriptionField] = Convert(Varchar(500) ,
[GJDS].[GJSourceDetail])
                                    , [EntryCount] = Count(1)
                            From      [Lookups].[GenJournalDetailSource] [GJDS]
                            Group By  [GJDS].[GJSource]
                                    , [GJDS].[GJSourceDetail]
                            Having    Count(1) > 1
                            Union
                            Select    [TableName] = '[Lookups].[GenJournalType]'
                                    , [Company] = 'ALL'
                                    , [ID] = Convert(Varchar(150) , [GJT].[TypeCode])
                                    , [DescriptionField] = Convert(Varchar(500) ,
[GJT].[TypeDetail])
                                    , [EntryCount] = Count(1)
                            From      [Lookups].[GenJournalType] [GJT]
                            Group By  [GJT].[TypeCode]
                                    , [GJT].[TypeDetail]
                            Having    Count(1) > 1
                            Union
                            Select    [TableName] = '[Lookups].[GenTransactionSource]'
                                    , [Company] = 'ALL'
                                    , [ID] = Convert(Varchar(150) , [GTS].[Source])
                                    , [DescriptionField] = Convert(Varchar(500) ,
[GTS].[SourceDesc])
                                    , [EntryCount] = Count(1)
                            From      [Lookups].[GenTransactionSource] [GTS]
                            Group By  [GTS].[Source]
                                    , [GTS].[SourceDesc]
                            Having    Count(1) > 1
                            Union
                            Select    [TableName] = '[Lookups].[GlAnalysisCategory]'
                                    , [Company] = [GAC].[Company]
                                    , [ID] = Convert(Varchar(150) , [GAC].[GlAnalysis-
Category])
                                    , [DescriptionField] = Convert(Varchar(500) ,
Null)
                                    , [EntryCount] = Count(1)
                            From      [Lookups].[GlAnalysisCategory] [GAC]
                            Group By  [GAC].[Company]
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Review.UspResults_-LookupDuplicates

```
                                    , [GAC].[GlAnalysisCategory]
                            Having    Count(1) > 1
                            Union
                            Select    [TableName] = '[Lookups].[GlExpenseCode]'
                                    , [Company] = [GEC].[Company]
                                    , [ID] = Convert(Varchar(150) , [GEC].[GlExpense-
Code])
                                    , [DescriptionField] = Convert(Varchar(500) ,
[GEC].[GlExpenseDescription])
                                    , [EntryCount] = Count(1)
                            From      [Lookups].[GlExpenseCode] [GEC]
                            Group By  [GEC].[Company]
                                    , [GEC].[GlExpenseCode]
                                    , [GEC].[GlExpenseDescription]
                            Having    Count(1) > 1
                            Union
                            Select    [TableName] = '[Lookups].[HolidayDays]'
                                    , [Company] = 'ALL'
                                    , [ID] = [HD].[Country] + ' - '
                                      + Convert(Varchar(150) , [HD].[HolidayDate])
                                    , [DescriptionField] = Convert(Varchar(500) ,
[HD].[HolidayDesc])
                                    , [EntryCount] = Count(1)
                            From      [Lookups].[HolidayDays] [HD]
                            Group By  [HD].[Country] + ' - '
                                      + Convert(Varchar(150) , [HD].[HolidayDate])
                                    , [HD].[HolidayDesc]
                            Having    Count(1) > 1
                            Union
                            Select    [TableName] = '[Lookups].[InvMaster_Part-
Category]'
                                    , [Company] = 'ALL'
                                    , [ID] = Convert(Varchar(150) , [IMPC].[Part-
CategoryCode])
                                    , [DescriptionField] = Convert(Varchar(500) ,
[IMPC].[PartCategoryDescription])
                                    , [EntryCount] = Count(1)
                            From      [Lookups].[InvMaster_PartCategory] [IMPC]
                            Group By  [IMPC].[PartCategoryCode]
                                    , [IMPC].[PartCategoryDescription]
                            Having    Count(1) > 1
                            Union
                            Select    [TableName] = '[Lookups].[JnlPostingType]'
                                    , [Company] = 'ALL'
                                    , [ID] = Convert(Varchar(150) , [JPT].[JnlPosting-
Type])
                                    , [DescriptionField] = Convert(Varchar(500) ,
[JPT].[JnlPostingTypeDesc])
                                    , [EntryCount] = Count(1)
                            From      [Lookups].[JnlPostingType] [JPT]
                            Group By  [JPT].[JnlPostingType]
                                    , [JPT].[JnlPostingTypeDesc]
                            Having    Count(1) > 1
                            Union
                            Select    [TableName] = '[Lookups].[JnlStatus]'
                                    , [Company] = 'ALL'
                                    , [ID] = Convert(Varchar(150) , [JS].[JnlStatus])
                                    , [DescriptionField] = Convert(Varchar(500) ,
[JS].[JnlStatusDesc])
```

Author: Johnson, Chris

```
                                , [EntryCount] = Count(1)
                        From      [Lookups].[JnlStatus] [JS]
                        Group By  [JS].[JnlStatus]
                                , [JS].[JnlStatusDesc]
                        Having    Count(1) > 1
                        Union
                /*Select    [TableName] = '[Lookups].[LedgerDepts]'
                        , [Company] = [LD].[Company]
                        , [ID] = Convert(Varchar(150) , [LD].[Department])
                        , [DescriptionField] = Convert(Varchar(500) ,
[LD].[DepartmentName])
                        , [EntryCount] = Count(1)
                From      [Lookups].[LedgerDepts] [LD]
                Group By  [LD].[Company]
                        , [LD].[Department]
                        , [LD].[DepartmentName]
                Having    Count(1) > 1
                Union*/
                        Select    [TableName] = '[Lookups].[LotTransactionTrn-
Type]'
                                , [Company] = 'ALL'
                                , [ID] = Convert(Varchar(150) , [LTTT].[TrnType])
                                , [DescriptionField] = Convert(Varchar(500) ,
[LTTT].[TrnTypeDescription])
                                , [EntryCount] = Count(1)
                        From      [Lookups].[LotTransactionTrnType] [LTTT]
                        Group By  [LTTT].[TrnType]
                                , [LTTT].[TrnTypeDescription]
                        Having    Count(1) > 1
                        Union
                        Select    [TableName] = '[Lookups].[MCompleteFlag]'
                                , [Company] = [MCF].[Company]
                                , [ID] = Convert(Varchar(150) , [MCF].[MComplete-
FlagCode])
                                , [DescriptionField] = Convert(Varchar(500) ,
[MCF].[MCompleteFlagDescription])
                                , [EntryCount] = Count(1)
                        From      [Lookups].[MCompleteFlag] [MCF]
                        Group By  [MCF].[Company]
                                , [MCF].[MCompleteFlagCode]
                                , [MCF].[MCompleteFlagDescription]
                        Having    Count(1) > 1
                        Union
                        Select    [TableName] = '[Lookups].[PorLineType]'
                                , [Company] = 'ALL'
                                , [ID] = Convert(Varchar(150) , [PLT].[PorLine-
Type])
                                , [DescriptionField] = Convert(Varchar(500) ,
[PLT].[PorLineTypeDesc])
                                , [EntryCount] = Count(1)
                        From      [Lookups].[PorLineType] [PLT]
                        Group By  [PLT].[PorLineType]
                                , [PLT].[PorLineTypeDesc]
                        Having    Count(1) > 1
                        Union
                        Select    [TableName] = '[Lookups].[ProductClass]'
                                , [Company] = [PC].[Company]
                                , [ID] = Convert(Varchar(150) , [PC].[Product-
Class])
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Review.UspResults_-LookupDuplicates

```
                                    , [DescriptionField] = Convert(Varchar(500) ,
[PC].[ProductClassDescription])
                                    , [EntryCount] = Count(1)
                            From      [Lookups].[ProductClass] [PC]
                            Group By  [PC].[Company]
                                    , [PC].[ProductClass]
                                    , [PC].[ProductClassDescription]
                            Having    Count(1) > 1
                            Union
                            Select    [TableName] = '[Lookups].[PurchaseOrderStatus]'
                                    , [Company] = [POS].[Company]
                                    , [ID] = Convert(Varchar(150) , [POS].[OrderStatus-
Code])
                                    , [DescriptionField] = Convert(Varchar(500) ,
[POS].[OrderStatusDescription])
                                    , [EntryCount] = Count(1)
                            From      [Lookups].[PurchaseOrderStatus] [POS]
                            Group By  [POS].[Company]
                                    , [POS].[OrderStatusCode]
                                    , [POS].[OrderStatusDescription]
                            Having    Count(1) > 1
                            Union
                            Select    [TableName] = '[Lookups].[PurchaseOrderTax-
Status]'
                                    , [Company] = [POTS].[Company]
                                    , [ID] = Convert(Varchar(150) , [POTS].[TaxStatus-
Code])
                                    , [DescriptionField] = Convert(Varchar(500) ,
[POTS].[TaxStatusDescription])
                                    , [EntryCount] = Count(1)
                            From      [Lookups].[PurchaseOrderTaxStatus] [POTS]
                            Group By  [POTS].[Company]
                                    , [POTS].[TaxStatusCode]
                                    , [POTS].[TaxStatusDescription]
                            Having    Count(1) > 1
                            Union
                            Select    [TableName] = '[Lookups].[PurchaseOrderType]'
                                    , [Company] = [POT].[Company]
                                    , [ID] = Convert(Varchar(150) , [POT].[OrderType-
Code])
                                    , [DescriptionField] = Convert(Varchar(500) ,
[POT].[OrderTypeDescription])
                                    , [EntryCount] = Count(1)
                            From      [Lookups].[PurchaseOrderType] [POT]
                            Group By  [POT].[Company]
                                    , [POT].[OrderTypeCode]
                                    , [POT].[OrderTypeDescription]
                            Having    Count(1) > 1
                            Union
                            Select    [TableName] = '[Lookups].[ReqnStatus]'
                                    , [Company] = [RS].[Company]
                                    , [ID] = Convert(Varchar(150) , [RS].[ReqnStatus-
Code])
                                    , [DescriptionField] = Convert(Varchar(500) ,
[RS].[ReqnStatusDescription])
                                    , [EntryCount] = Count(1)
                            From      [Lookups].[ReqnStatus] [RS]
                            Group By  [RS].[Company]
                                    , [RS].[ReqnStatusCode]
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Review.UspResults_-LookupDuplicates

```
                                    , [RS].[ReqnStatusDescription]
                            Having    Count(1) > 1
                            Union
                            Select    [TableName] = '[Lookups].[SalesOrderLineType]'
                                    , [Company] = [SOLT].[Company]
                                    , [ID] = Convert(Varchar(150) , [SOLT].[LineType-
Code])
                                    , [DescriptionField] = Convert(Varchar(500) ,
[SOLT].[LineTypeDescription])
                                    , [EntryCount] = Count(1)
                            From      [Lookups].[SalesOrderLineType] [SOLT]
                            Group By  [SOLT].[Company]
                                    , [SOLT].[LineTypeCode]
                                    , [SOLT].[LineTypeDescription]
                            Having    Count(1) > 1
                            Union
                            Select    [TableName] = '[Lookups].[SalesOrderStatus]'
                                    , [Company] = [SOS].[Company]
                                    , [ID] = Convert(Varchar(150) , [SOS].[OrderStatus-
Code])
                                    , [DescriptionField] = Convert(Varchar(500) ,
[SOS].[OrderStatusDescription])
                                    , [EntryCount] = Count(1)
                            From      [Lookups].[SalesOrderStatus] [SOS]
                            Group By  [SOS].[Company]
                                    , [SOS].[OrderStatusCode]
                                    , [SOS].[OrderStatusDescription]
                            Having    Count(1) > 1
                            Union
                            Select    [TableName] = '[Lookups].[TrnTypeAmount-
Modifier]'
                                    , [Company] = [TTAM].[Company]
                                    , [ID] = Convert(Varchar(150) , [TTAM].[TrnType])
                                    , [DescriptionField] = Convert(Varchar(500) ,
[TTAM].[AmountModifier])
                                    , [EntryCount] = Count(1)
                            From      [Lookups].[TrnTypeAmountModifier] [TTAM]
                            Group By  [TTAM].[Company]
                                    , [TTAM].[TrnType]
                                    , [TTAM].[AmountModifier]
                            Having    Count(1) > 1
                            Union
                            Select    [TableName] = '[Lookups].[Warehouse]'
                                    , [Company] = [W].[Company]
                                    , [ID] = [W].[Warehouse]
                                    , [DescriptionField] = Convert(Varchar(500) ,
[W].[WarehouseDescription])
                                    , [EntryCount] = Count(1)
                            From      [Lookups].[Warehouse] [W]
                            Group By  [W].[Company]
                                    , [W].[Warehouse]
                                    , [W].[WarehouseDescription]
                            Having    Count(1) > 1
                          ) [t];

        Set NoCount Off;
        Select  [DC].[TableName]
              , [DC].[Company]
              , [DC].[ID]
```

Author: Johnson, Chris

```
                , [DC].[DescriptionField]
                , [DC].[DupeCount]
        From    [#DupeCheck] [DC];

        Set NoCount On;
        Drop Table [#DupeCheck];
    End;
GO
EXEC sp_addextendedproperty N'MS_Description', N'check lookup tables for duplicates
(indicates that refresh failed)', 'SCHEMA', N'Review', 'PROCEDURE', N'UspResults_-
LookupDuplicates', NULL, NULL
GO
```

Uses

[Lookups].[AdmTransactionIDs]
[Lookups].[ApJnlDistribExpenseType]
[Lookups].[ArInvoicePayTrnType]
[Lookups].[BankBalances]
[Lookups].[Bin]
[Lookups].[BudgetType]
[Lookups].[CompanyNames]
[Lookups].[CurrencyRates]
[Lookups].[GenJournalCtlJnlSource]
[Lookups].[GenJournalCtlSource]
[Lookups].[GenJournalDetailSource]
[Lookups].[GenJournalType]
[Lookups].[GenTransactionSource]
[Lookups].[GlAnalysisCategory]
[Lookups].[GlExpenseCode]
[Lookups].[HolidayDays]
[Lookups].[InvMaster_PartCategory]
[Lookups].[JnlPostingType]
[Lookups].[JnlStatus]
[Lookups].[LotTransactionTrnType]
[Lookups].[MCompleteFlag]
[Lookups].[PorLineType]
[Lookups].[ProductClass]
[Lookups].[PurchaseOrderStatus]
[Lookups].[PurchaseOrderTaxStatus]
[Lookups].[PurchaseOrderType]
[Lookups].[ReqnStatus]
[Lookups].[SalesOrderLineType]
[Lookups].[SalesOrderStatus]
[Lookups].[TrnTypeAmountModifier]
[Lookups].[Warehouse]
[Process].[UspInsert_RedTagLogs]
[Review]

Author: Johnson, Chris

[Review].[UspResults_ProcsMostRun]

MS_Description

details of the most commonly used stored procedures

Parameters

Name	Data Type	Max Length (Bytes)
@RedTagType	char	1
@RedTagUse	varchar(500)	500

SQL Script

```
CREATE Proc [Review].[UspResults_ProcsMostRun]
    (
      @RedTagType Char(1)
    , @RedTagUse Varchar(500)
    )
As
    Begin
        Set NoCount On;

    --Red tag

        Declare @RedTagDB Varchar(255)= Db_Name();
        Exec [Process].[UspInsert_RedTagLogs] @StoredProcDb = 'BlackBox' ,
            @StoredProcSchema = 'Review' ,
            @StoredProcName = 'UspResults_ProcsMostRun' ,
            @UsedByType = @RedTagType , @UsedByName = @RedTagUse ,
            @UsedByDb = @RedTagDB;

        Create Table [#ProcsRun_NonDev]
            (
              [ProcName] Varchar(500) Collate Latin1_General_BIN
            , [SchemaName] Varchar(150) Collate Latin1_General_BIN
            , [RunTime] DateTime
            , [SPs] Int
            );


        Insert  [#ProcsRun_NonDev]
                ( [ProcName]
                , [SchemaName]
                , [RunTime]
                , [SPs]
                )
                Select  [RTL].[StoredProcName]
                      , [RTL].[StoredProcSchema]
                      , [RunTime] = Convert(DateTime , ( Left(Convert(Varchar(255) ,
[RTL].[TagDatetime]) ,
                                                              18) + '0' ))
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Review.UspResults_-ProcsMostRun

```
                      , [SPs] = Count([RTL].[TagID])
                From    [BlackBox].[History].[RedTagLogs] [RTL]
                        Left Join [Lookups].[RedTagsUsedByType] [RTUBT]
                            On [RTUBT].[UsedByType] = [RTL].[UsedByType]
                Where   [RTL].[UsedByName] Not Like 'Development%'
                        And [RTL].[UsedByName] Not Like 'Chris Johnson%'
                Group By Convert(DateTime , ( Left(Convert(Varchar(255) , [RTL].[Tag-
Datetime]) ,
                                                   18) + '0' ))
                      , [RTL].[StoredProcName]
                      , [RTL].[StoredProcSchema];

        Set NoCount Off;
        Select  [RR].[ProcName]
              , [RR].[SchemaName]
              , [24Hours] = Count(Distinct Case When DateDiff(Day ,
                                                              [RR].[RunTime] ,
                                                              GetDate()) <= 1
                                                Then [RR].[RunTime]
                                                Else Null
                                           End)
              , [7Days] = Count(Distinct Case When DateDiff(Day ,
                                                            [RR].[RunTime] ,
                                                            GetDate()) Between 2 And
7
                                              Then [RR].[RunTime]
                                              Else Null
                                         End)
              , [1Month] = Count(Distinct Case When DateDiff(Day ,
                                                             [RR].[RunTime] ,
                                                             GetDate()) > 7
                                                And DateDiff(Month ,
                                                             [RR].[RunTime] ,
                                                             GetDate()) < 1
                                               Then [RR].[RunTime]
                                               Else Null
                                          End)
              , [1MonthPlus] = Count(Distinct Case When DateDiff(Month ,
                                                              [RR].[RunTime] ,
                                                              GetDate()) >= 1
                                                  Then [RR].[RunTime]
                                                  Else Null
                                             End)
              , [TotalRuns] = Count(Distinct [RR].[RunTime])
              , [Ranking] = Count(Distinct Case When DateDiff(Day ,
                                                              [RR].[RunTime] ,
                                                              GetDate()) <= 1
                                                Then [RR].[RunTime]
                                                Else Null
                                           End) * 4--24Hours

                + Count(Distinct Case When DateDiff(Day , [RR].[RunTime] ,
                                                    GetDate()) Between 2 And 7
                                      Then [RR].[RunTime]
                                      Else Null
                                 End) * 3--1 week
```

Author: Johnson, Chris

```
                + Count(Distinct Case When DateDiff(Day , [RR].[RunTime] ,
                                                    GetDate()) > 7
                                          And DateDiff(Month , [RR].[RunTime] ,
                                                       GetDate()) < 1
                                     Then [RR].[RunTime]
                                     Else Null
                                End) * 2--1 month

                + Count(Distinct Case When DateDiff(Month , [RR].[RunTime] ,
                                                    GetDate()) >= 1
                                     Then [RR].[RunTime]
                                     Else Null
                                End)--1month plus

        From    [#ProcsRun_NonDev] [RR]
        Group By [RR].[ProcName]
              , [RR].[SchemaName]
        Order By [Ranking] Desc;

        Set NoCount On;
        Drop Table [#ProcsRun_NonDev];
    End;
GO
EXEC sp_addextendedproperty N'MS_Description', N'details of the most commonly used
stored procedures', 'SCHEMA', N'Review', 'PROCEDURE', N'UspResults_ProcsMostRun',
NULL, NULL
GO
```

Uses

[History].[RedTagLogs]
[Lookups].[RedTagsUsedByType]
[Process].[UspInsert_RedTagLogs]
[Review]

Author: Johnson, Chris

[Review].[UspResults_ReportsDevelopmentRun]

MS_Description

details of reports in development run

Parameters

Name	Data Type	Max Length (Bytes)
@RedTagType	char	1
@RedTagUse	varchar(500)	500

SQL Script

```
CREATE Proc [Review].[UspResults_ReportsDevelopmentRun]
    (
      @RedTagType Char(1)
    , @RedTagUse Varchar(500)
    )
As
    Begin
        Set NoCount On;

    --Red tag

        Declare @RedTagDB Varchar(255)= Db_Name();
        Exec [Process].[UspInsert_RedTagLogs] @StoredProcDb = 'BlackBox' ,
            @StoredProcSchema = 'Review' ,
            @StoredProcName = 'UspResults_ReportsDevelopmentRun' ,
            @UsedByType = @RedTagType , @UsedByName = @RedTagUse ,
            @UsedByDb = @RedTagDB;

        Create Table [#ReportRuns_NonDev]
            (
              [ReportName] Varchar(500) Collate Latin1_General_BIN
            , [ReportType] Varchar(150) Collate Latin1_General_BIN
            , [RunTime] DateTime
            , [SPs] Int
            );


        Insert  [#ReportRuns_NonDev]
                ( [ReportName]
                , [ReportType]
                , [RunTime]
                , [SPs]
                )
                Select  [RTL].[UsedByName]
                      , [RTUBT].[UsedByDescription]
                      , [RunTime] = Convert(DateTime , ( Left(Convert(Varchar(255) ,
[RTL].[TagDatetime]) ,
                                                              18) + '0' ))
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Review.UspResults_-ReportsDevelopmentRun

```
                      , [SPs] = Count([RTL].[TagID])
                From    [BlackBox].[History].[RedTagLogs] [RTL]
                        Left Join [Lookups].[RedTagsUsedByType] [RTUBT]
                            On [RTUBT].[UsedByType] = [RTL].[UsedByType]
                Where   [RTL].[UsedByName] Like 'Development%'
                        Or [RTL].[UsedByName] Like 'Chris Johnson%'
                Group By Convert(DateTime , ( Left(Convert(Varchar(255) , [RTL].[Tag-
Datetime]) ,
                                                   18) + '0' ))
                      , [RTL].[UsedByName]
                      , [RTUBT].[UsedByDescription];

        Set NoCount Off;
        Select  [RR].[ReportName]
              , [RR].[ReportType]
              , [24Hours] = Count(Distinct Case When DateDiff(Day ,
                                                              [RR].[RunTime] ,
                                                              GetDate()) <= 1
                                                Then [RR].[RunTime]
                                                Else Null
                                           End)
              , [7Days] = Count(Distinct Case When DateDiff(Day ,
                                                            [RR].[RunTime] ,
                                                            GetDate()) Between 2 And
7
                                              Then [RR].[RunTime]
                                              Else Null
                                         End)
              , [1Month] = Count(Distinct Case When DateDiff(Day ,
                                                             [RR].[RunTime] ,
                                                             GetDate()) > 7
                                                And DateDiff(Month ,
                                                             [RR].[RunTime] ,
                                                             GetDate()) <= 1
                                               Then [RR].[RunTime]
                                               Else Null
                                          End)
              , [1MonthPlus] = Count(Distinct Case When DateDiff(Month ,
                                                              [RR].[RunTime] ,
                                                              GetDate()) > 1
                                                   Then [RR].[RunTime]
                                                   Else Null
                                              End)
              , [TotalRuns] = Count(Distinct [RR].[RunTime])
              , [Ranking] = Count(Distinct Case When DateDiff(Day ,
                                                              [RR].[RunTime] ,
                                                              GetDate()) <= 1
                                                Then [RR].[RunTime]
                                                Else Null
                                           End) * 4--24Hours

                + Count(Distinct Case When DateDiff(Day , [RR].[RunTime] ,
                                                    GetDate()) Between 2 And 7
                                      Then [RR].[RunTime]
                                      Else Null
                                 End) * 3--1 week
```

Author: Johnson, Chris

```
                + Count(Distinct Case When DateDiff(Day , [RR].[RunTime] ,
                                                    GetDate()) > 7
                                          And DateDiff(Month , [RR].[RunTime] ,
                                                       GetDate()) <= 1
                                     Then [RR].[RunTime]
                                     Else Null
                                End) * 2--1 month

                + Count(Distinct Case When DateDiff(Month , [RR].[RunTime] ,
                                                    GetDate()) > 1
                                     Then [RR].[RunTime]
                                     Else Null
                                End)--1month plus

        From    [#ReportRuns_NonDev] [RR]
        Group By [RR].[ReportName]
              , [RR].[ReportType]
        Order By [Ranking] Desc;

        Set NoCount On;
        Drop Table [#ReportRuns_NonDev];
    End;
GO
EXEC sp_addextendedproperty N'MS_Description', N'details of reports in development
run', 'SCHEMA', N'Review', 'PROCEDURE', N'UspResults_ReportsDevelopmentRun', NULL,
NULL
GO
```

Uses

[History].[RedTagLogs]
[Lookups].[RedTagsUsedByType]
[Process].[UspInsert_RedTagLogs]
[Review]

Author: Johnson, Chris

[Review].[UspResults_ReportsMostRun]

MS_Description

details of the most commonly used reports

Parameters

Name	Data Type	Max Length (Bytes)
@RedTagType	char	1
@RedTagUse	varchar(500)	500

SQL Script

```
CREATE Proc [Review].[UspResults_ReportsMostRun]
    (
      @RedTagType Char(1)
    , @RedTagUse Varchar(500)
    )
As
    Begin
        Set NoCount On;

    --Red tag

        Declare @RedTagDB Varchar(255)= Db_Name();
        Exec [Process].[UspInsert_RedTagLogs] @StoredProcDb = 'BlackBox' ,
            @StoredProcSchema = 'Review' ,
            @StoredProcName = 'UspResults_ReportsMostRun' ,
            @UsedByType = @RedTagType , @UsedByName = @RedTagUse ,
            @UsedByDb = @RedTagDB;

        Create Table [#ReportRuns_NonDev]
            (
              [ReportName] Varchar(500) Collate Latin1_General_BIN
            , [ReportType] Varchar(150) Collate Latin1_General_BIN
            , [RunTime] DateTime
            , [SPs] Int
            );


        Insert  [#ReportRuns_NonDev]
                ( [ReportName]
                , [ReportType]
                , [RunTime]
                , [SPs]
                )
                Select  [RTL].[UsedByName]
                      , [RTUBT].[UsedByDescription]
                      , [RunTime] = Convert(DateTime , ( Left(Convert(Varchar(255) ,
[RTL].[TagDatetime]) ,
                                                              18) + '0' ))
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Review.UspResults_-ReportsMostRun

```
                      , [SPs] = Count([RTL].[TagID])
                From    [BlackBox].[History].[RedTagLogs] [RTL]
                        Left Join [Lookups].[RedTagsUsedByType] [RTUBT]
                            On [RTUBT].[UsedByType] = [RTL].[UsedByType]
                Where   [RTL].[UsedByName] Not Like 'Development%'
                        And [RTL].[UsedByName] Not Like 'Chris Johnson%'
                Group By Convert(DateTime , ( Left(Convert(Varchar(255) , [RTL].[Tag-
Datetime]) ,
                                                   18) + '0' ))
                      , [RTL].[UsedByName]
                      , [RTUBT].[UsedByDescription];

        Set NoCount Off;
        Select  [RR].[ReportName]
              , [RR].[ReportType]
              , [24Hours] = Count(Distinct Case When DateDiff(Day ,
                                                              [RR].[RunTime] ,
                                                              GetDate()) <= 1
                                                Then [RR].[RunTime]
                                                Else Null
                                           End)
              , [7Days] = Count(Distinct Case When DateDiff(Day ,
                                                            [RR].[RunTime] ,
                                                            GetDate()) Between 2 And
7
                                              Then [RR].[RunTime]
                                              Else Null
                                         End)
              , [1Month] = Count(Distinct Case When DateDiff(Day ,
                                                             [RR].[RunTime] ,
                                                             GetDate()) > 7
                                                And DateDiff(Month ,
                                                             [RR].[RunTime] ,
                                                             GetDate()) < 1
                                               Then [RR].[RunTime]
                                               Else Null
                                          End)
              , [1MonthPlus] = Count(Distinct Case When DateDiff(Month ,
                                                              [RR].[RunTime] ,
                                                              GetDate()) >= 1
                                                  Then [RR].[RunTime]
                                                  Else Null
                                             End)
              , [TotalRuns] = Count(Distinct [RR].[RunTime])
              , [Ranking] = Count(Distinct Case When DateDiff(Day ,
                                                              [RR].[RunTime] ,
                                                              GetDate()) <= 1
                                                Then [RR].[RunTime]
                                                Else Null
                                           End) * 4--24Hours

                + Count(Distinct Case When DateDiff(Day , [RR].[RunTime] ,
                                                    GetDate()) Between 2 And 7
                                      Then [RR].[RunTime]
                                      Else Null
                                 End) * 3--1 week
```

Author: Johnson, Chris

```
                + Count(Distinct Case When DateDiff(Day , [RR].[RunTime] ,
                                                    GetDate()) > 7
                                          And DateDiff(Month , [RR].[RunTime] ,
                                                       GetDate()) < 1
                                     Then [RR].[RunTime]
                                     Else Null
                                End) * 2--1 month

                + Count(Distinct Case When DateDiff(Month , [RR].[RunTime] ,
                                                    GetDate()) >= 1
                                     Then [RR].[RunTime]
                                     Else Null
                                End)--1month plus

        From    [#ReportRuns_NonDev] [RR]
        Group By [RR].[ReportName]
              , [RR].[ReportType]
        Order By [Ranking] Desc;

        Set NoCount On;
        Drop Table [#ReportRuns_NonDev];
    End;
GO
EXEC sp_addextendedproperty N'MS_Description', N'details of the most commonly used
reports', 'SCHEMA', N'Review', 'PROCEDURE', N'UspResults_ReportsMostRun', NULL, NULL
GO
```

Uses

[History].[RedTagLogs]
[Lookups].[RedTagsUsedByType]
[Process].[UspInsert_RedTagLogs]
[Review]

Author: Johnson, Chris

[Review].[UspResults_SysproTransactionsCheck]

MS_Description

check of non-entered audit logs into history tables

Parameters

Name	Data Type	Max Length (Bytes)
@RedTagType	char	1
@RedTagUse	varchar(500)	500

SQL Script

```
Create Proc [Review].[UspResults_SysproTransactionsCheck]
    (
      @RedTagType Char(1)
    , @RedTagUse Varchar(500)
    )
As
    Begin
        Set NoCount On;

    --Red tag

        Declare @RedTagDB Varchar(255)= Db_Name();
        Exec [Process].[UspInsert_RedTagLogs] @StoredProcDb = 'BlackBox' ,
            @StoredProcSchema = 'Review' ,
            @StoredProcName = 'UspResults_SysproTransactionsCheck' ,
            @UsedByType = @RedTagType , @UsedByName = @RedTagUse ,
            @UsedByDb = @RedTagDB;

        Set NoCount Off;
        Select  [LastSignatureDateTime] = Max([STL].[SignatureDateTime])
              , [SumOfErrors] = Sum(Convert(Int , [STL].[IsError]))
              , [SumOfNonEntered] = Sum(1
                                        - Convert(Int , [STL].[AlreadyEntered]))
        From    [Process].[SysproTransactionsLogged] [STL];
    End;
GO
EXEC sp_addextendedproperty N'MS_Description', N'check of non-entered audit logs
into history tables', 'SCHEMA', N'Review', 'PROCEDURE', N'UspResults_Syspro-
TransactionsCheck', NULL, NULL
GO
```

Uses

[Process].[SysproTransactionsLogged]
[Process].[UspInsert_RedTagLogs]
[Review]

Author: Johnson, Chris

[Review].[UspResults_SysproTransactionsTrending]

MS_Description

review of syspro audit logs captured in BlackBox

Parameters

Name	Data Type	Max Length (Bytes)
@RedTagType	char	1
@RedTagUse	varchar(500)	500

SQL Script

```
Create Proc [Review].[UspResults_SysproTransactionsTrending]
    (
      @RedTagType Char(1)
    , @RedTagUse Varchar(500)
    )
As
    Begin
        Set NoCount On;

    --Red tag

        Declare @RedTagDB Varchar(255)= Db_Name();
        Exec [Process].[UspInsert_RedTagLogs] @StoredProcDb = 'BlackBox' ,
            @StoredProcSchema = 'Review' ,
            @StoredProcName = 'UspResults_SysproTransactionsCheck' ,
            @UsedByType = @RedTagType , @UsedByName = @RedTagUse ,
            @UsedByDb = @RedTagDB;

        Set NoCount Off;
        Select  [MonthOfSignatures] = DateFromParts(DatePart(Year ,
                                                             [STL].[SignatureDate-
Time]) ,
                                                    DatePart(Month ,
                                                             [STL].[SignatureDate-
Time]) ,
                                                    1)
              , [CountOfLogs] = Count(Distinct Convert(Varchar(155) ,
[STL].[SignatureDateTime])
                                      + [STL].[TableName]
                                      + Coalesce([STL].[ConditionName] , '')
                                      + [STL].[ItemKey])
              , [CountOfTables] = Count(Distinct [STL].[TableName])
              , [CountOfConditions] = Count(Distinct [STL].[ConditionName])
              , [CountOfKeys] = Count(Distinct [STL].[ItemKey])
        From    [Process].[SysproTransactionsLogged] [STL]
        Where   DateDiff(Month ,
                         DateFromParts(DatePart(Year ,
                                                [STL].[SignatureDateTime]) ,
```

Author: Johnson, Chris

```
                                        DatePart(Month ,
                                                 [STL].[SignatureDateTime]) , 1) ,
                          DateFromParts(DatePart(Year , GetDate()) ,
                                        DatePart(Month , GetDate()) , 1)) <= 12
        Group By DateFromParts(DatePart(Year , [STL].[SignatureDateTime]) ,
                               DatePart(Month , [STL].[SignatureDateTime]) , 1)
        Order By DateFromParts(DatePart(Year , [STL].[SignatureDateTime]) ,
                               DatePart(Month , [STL].[SignatureDateTime]) , 1)
Desc;
    End;
GO
EXEC sp_addextendedproperty N'MS_Description', N'review of syspro audit logs
captured in BlackBox', 'SCHEMA', N'Review', 'PROCEDURE', N'UspResults_Syspro-
TransactionsTrending', NULL, NULL
GO
```

Uses

[Process].[SysproTransactionsLogged]
[Process].[UspInsert_RedTagLogs]
[Review]

Author: Johnson, Chris

[Review].[UspResults_WarehousesIssueFrom]

MS_Description

details of warehouse that do not have issue from completed

Parameters

Name	Data Type	Max Length (Bytes)
@IncludeNotes	bit	1
@RedTagType	char	1
@RedTagUse	varchar(500)	500

SQL Script

```
CREATE Proc [Review].[UspResults_WarehousesIssueFrom]
    (
      @IncludeNotes Bit
    , @RedTagType Char(1)
    , @RedTagUse Varchar(500)
    )
As
    Begin
        Set NoCount On;
        Select  @IncludeNotes = Coalesce(@IncludeNotes , 0);

    --Red tag

        Declare @RedTagDB Varchar(255)= Db_Name();
        Exec [Process].[UspInsert_RedTagLogs] @StoredProcDb = 'BlackBox' ,
            @StoredProcSchema = 'Review' ,
            @StoredProcName = 'UspResults_WarehousesIssueFrom' ,
            @UsedByType = @RedTagType , @UsedByName = @RedTagUse ,
            @UsedByDb = @RedTagDB;

        Create Table [#WHReview]
            (
              [IssueFrom] Varchar(150)            collate latin1_general_bin
            , [Warehouse] Varchar(50)            collate latin1_general_bin
            , [Description] Varchar(150)        collate latin1_general_bin
            , [DB] Varchar(150)                    collate latin1_general_bin
            , [RowType] Varchar(10)                collate latin1_general_bin
            );

        If @IncludeNotes = 1
            Begin
                Insert  [#WHReview]
                        ( [IssueFrom]
                        , [Warehouse]
                        , [Description]
                        , [DB]
```

Author: Johnson, Chris

```
                        , [RowType]
                        )
                Values  ( ''  -- IssueFrom - varchar(150)

                        , 'List of '  -- Warehouse - varchar(50)

                        , 'warehouses that '  -- Description - varchar(150)

                        , 'have not been completed with issue from data' -- DB -
varchar(150)

                        , 'Notes'  -- RowType - varchar(10)

                        );
            End;



        Exec [BlackBox].[Process].[ExecForEachDB] @cmd = N'Use [?];
If Lower(replace(''?'',''['','''')) Like ''sysprocompany%'' And Is-
Numeric(Right(replace(''?'','']'',''''),1))=1
BEGIN
Insert [#WHReview]
        ( [IssueFrom]
        , [Warehouse]
        , [Description]
        , [DB]
        , [RowType]
        )
Select  [IWC].[Fax]
      , [IWC].[Warehouse]
      , [IWC].[Description]
      , [DB] = ''?''
      , [RowType] = ''Data''
From    [dbo].[InvWhControl] [IWC];
end';


        Set NoCount Off;

        Select  [WR].[Warehouse]
              , [WR].[Description]
              , [WR].[DB]
              , [WR].[RowType]
        From    [#WHReview] [WR]
        Where   [WR].[IssueFrom] Not In ( 'Y' , 'N' )
        Order By [WR].[RowType] Desc;
    End;
GO
EXEC sp_addextendedproperty N'MS_Description', N'details of warehouse that do not
have issue from completed', 'SCHEMA', N'Review', 'PROCEDURE', N'UspResults_-
WarehousesIssueFrom', NULL, NULL
GO
```

Project> BORN> User databases> BlackBox> Programmability> Stored Procedures> Review.UspResults_-WarehousesIssueFrom

Uses

[Process].[ExecForEachDB]

[Process].[UspInsert_RedTagLogs]

[Review]

Author: Johnson, Chris

Table-valued Functions

Objects

Name
dbo.udf_SplitString function used to parse multiple values separated by a delimiter
dbo.UdfResults_NumberRange This function returns an integer table containing all integers in the range of @START_NUMBER through @END_-NUMBER, inclusive. The maximum number of rows that this function can return is 16777216.

Author: Johnson, Chris

[dbo].[udf_SplitString]

MS_Description

function used to parse multiple values separated by a delimiter

Parameters

Name	Data Type	Max Length (Bytes)
@DelimittedString	varchar(max)	max
@Delimiter	varchar	1

SQL Script

```
Create --drop --alter

Function [dbo].[udf_SplitString]
    (
      @DelimittedString [VARCHAR](Max) --Text to be split

      ,@Delimiter [VARCHAR](1)  --delimiter to be used

    )
/*
////////////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////
///            Function designed by Chris Johnson, Prometic Group September 2015
///
///
///
///            Function set out to create a table from a delimited text field for
use in stored procedures & SSRS                          ///
///
///
///
///
///            Version 1.0
///
///
///
///            Change Log
///
///
///
///            Date        Person                    Description
///
///            7/9/2015    Chris Johnson            Initial version created
///
///            ??/??/201?    Placeholder                Placeholder
///
///            ??/??/201?    Placeholder                Placeholder
///
////////////////////////////////////////////////////////////////////////////////////////
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Functions> Table-valued Functions> dbo.udf_-SplitString

```
////////////////////////////////////////////////
*/

--define return table

Returns @Table TABLE ( Value VARCHAR(250) )

    Begin

        --End text to be split with a comma

        Set @DelimittedString = COALESCE(@DelimittedString, '') + @Delimiter

        --iterate through the text, removing each field individually and trimming
the text field

        While LEN(@DelimittedString) > 0
            Begin
                Insert @Table (Value)
                        Select Value = SUBSTRING(@DelimittedString, 1,
                                    CHARINDEX(@Delimiter, @DelimittedString) - 1)

                Set @DelimittedString = RIGHT(@DelimittedString,
                                    LEN(@DelimittedString) - CHARINDEX(@Delimiter,
@DelimittedString))
            End;
        Return;
    End;




GO
EXEC sp_addextendedproperty N'MS_Description', N'function used to parse multiple
values separated by a delimiter', 'SCHEMA', N'dbo', 'FUNCTION', N'udf_SplitString',
NULL, NULL
GO
```

Used By

[Process].[ExecForEachDB_WithTableCheck]
[Process].[UspPopulate_UnpivotHistory]
[Report].[UspResults_RequisitionStatus]
[Report].[UspResults_StockMovements]

Author: Johnson, Chris

[dbo].[UdfResults_NumberRange]

MS_Description

This function returns an integer table containing all integers in the range of @START_NUMBER through @END_-NUMBER, inclusive. The maximum number of rows that this function can return is 16777216.

Parameters

Name	Data Type	Max Length (Bytes)
@StartNumber	int	4
@EndNumber	int	4

SQL Script

```
Create Function [dbo].[UdfResults_NumberRange]
    (
      @StartNumber Int
    , @EndNumber Int
    )
/*
This function returns an integer table containing all integers in the range
of@START_NUMBER through @END_NUMBER, inclusive. The maximum number of rows that this
function can return is 16777216.
Original Query - http://www.sqlteam.com/forums/topic.asp?TOPIC_ID=47685&Search-
Terms=F_TABLE_NUMBER_RANGE
*/
Returns Table
As

Return
    ( Select [Number] = ( [a].[Number] + [b].[Number] )
                + -- Add the starting number for the final result set - The case is
needed, because the start and end - numbers can be passed in any order

    Case When @StartNumber <= @EndNumber Then @StartNumber
         Else @EndNumber
    End
      From      ( Select Top 100 Percent
                            [Number] = Convert(Int , [N1].[N01] + [N2].[N02]
                            + [N3].[N03])
                  From      -- Cross rows from 3 tables based on powers of 16 -
Maximum number of rows from cross join is 4096, 0 to 4095

                            ( Select    [N01] = 0
                              Union All Select    1 Union All Select    2
                              Union All Select    3 Union All Select    4
                              Union All Select    5 Union All Select    6
                              Union All Select    7 Union All Select    8
                              Union All Select    9 Union All Select    10
                              Union All Select    11 Union All Select    12
                              Union All Select    13 Union All Select    14
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Functions> Table-valued Functions> dbo.Udf-Results_NumberRange

```
                              Union All Select    15 ) [N1]
                            Cross Join ( Select [N02] = 0
                                         Union All Select 16 Union All Select 32
                                         Union All Select 48 Union All Select 64
                                         Union All Select 80 Union All Select 96
                                         Union All Select 112 Union All Select 128
                                         Union All Select 144 Union All Select 160
                                         Union All Select 176 Union All Select 192
                                         Union All Select 208 Union All Select 224
                                         Union All Select 240 ) [N2]
                            Cross Join ( Select [N03] = 0
                                         Union All Select 256 Union All Select 512
                                         Union All Select 768 Union All Select 1024
                                         Union All Select 1280 Union All Select 1536
                                         Union All Select 1792 Union All Select 2048
                                         Union All Select 2304 Union All Select 2560
                                         Union All Select 2816 Union All Select 3072
                                         Union All Select 3328 Union All Select 3584
                                         Union All Select 3840 ) [N3]
                    Where     -- Minimize the number of rows crossed by selecting only
rows - with a value less the the square root of rows needed.

                              [N1].[N01] + [N2].[N02] + [N3].[N03] < -- Square root of
total rows rounded up to next whole number

            Convert(Int , Ceiling(Sqrt(Abs(@StartNumber - @EndNumber) + 1)))
                      Order By  1
                   ) [a]
                   Cross Join ( Select Top 100 Percent
                                       [Number] = Convert(Int , ( [N1].[N01]
                                                                + [N2].[N02]
                                                                + [N3].[N03] )
                                       * -- Square root of total rows rounded up to
next whole number

            Convert(Int , Ceiling(Sqrt(Abs(@StartNumber - @EndNumber) + 1))))
                                From   -- Cross rows from 3 tables based on powers of
16 - Maximum number of rows from cross join is 4096, 0 to 4095

                                       ( Select    [N01] = 0
                                         Union All Select    1 Union All Select    2
                                         Union All Select    3 Union All Select    4
                                         Union All Select    5 Union All Select    6
                                         Union All Select    7 Union All Select    8
                                         Union All Select    9 Union All Select    10
                                         Union All Select    11 Union All Select    12
                                         Union All Select    13 Union All Select    14
                                         Union All Select    15 ) [N1]
                                       Cross Join ( Select [N02] = 0
                                                    Union All Select 16 Union All
Select 32
                                                    Union All Select 48 Union All
Select 64
                                                    Union All Select 80 Union All
Select 96
                                                    Union All Select 112 Union All
Select 128
                                                    Union All Select 144 Union All
Select 160
```

Author: Johnson, Chris

```
                                                Union All Select 176 Union All
Select 192
                                                Union All Select 208 Union All
Select 224
                                                Union All Select 240 ) [N2]
                                    Cross Join ( Select [N03] = 0
                                                Union All Select 256 Union All
Select 512
                                                Union All Select 768 Union All
Select 1024
                                                Union All Select 1280 Union All
Select 1536
                                                Union All Select 1792 Union All
Select 2048
                                                Union All Select 2304 Union All
Select 2560
                                                Union All Select 2816 Union All
Select 3072
                                                Union All Select 3328 Union All
Select 3584
                                                Union All Select 3840 ) [N3]
                            Where  -- Minimize the number of rows crossed by
selecting only rows - with a value less the the square root of rows needed.

                                    [N1].[N01] + [N2].[N02] + [N3].[N03] < -- Square
root of total rows rounded up to next whole number

        Convert(Int , Ceiling(Sqrt(Abs(@StartNumber - @EndNumber) + 1)))
                            Order By 1
                          ) [b]
      Where     [a].[Number] + [b].[Number] < -- Total number of rows

    Abs(@StartNumber - @EndNumber) + 1
                And
    -- Check that the number of rows to be returned - is less than or equal to the
maximum of 16777216

                Case When Abs(@StartNumber - @EndNumber) + 1 <= 16777216
                     Then 1
                     Else 0
                End = 1
    );


GO
EXEC sp_addextendedproperty N'MS_Description', N'This function returns an integer
table containing all integers in the range of @START_NUMBER through @END_NUMBER,
inclusive. The maximum number of rows that this function can return is 16777216.',
'SCHEMA', N'dbo', 'FUNCTION', N'UdfResults_NumberRange', NULL, NULL
GO
```

Used By

[Report].[UspResults_GLMovements]
[Report].[UspResults_GLMovementsCenter]

Author: Johnson, Chris

Scalar-valued Functions

Objects

Name
Process.Udf_WorkingDays generates the number of working days between two dates

Author: Johnson, Chris

[Process].[Udf_WorkingDays]

MS_Description

generates the number of working days between two dates

Parameters

Name	Data Type	Max Length (Bytes)
@StartDate	date	3
@EndDate	date	3
@Country	varchar(150)	150

SQL Script

```
CREATE Function [Process].[Udf_WorkingDays]
    (
      @StartDate Date
    , @EndDate Date
    , @Country Varchar(150)
    )
Returns Int
As
    Begin
        Declare @DaysToExclude Int, @CurrentDate Date=@StartDate;
        Declare @DaysCount Table (CalendarDate Date);


        While @CurrentDate<=@EndDate
        Begin
            If DateName(Weekday,@CurrentDate) Not In ('Saturday','Sunday') And Not
Exists (Select 1 From [Lookups].[HolidayDays] As [HD] Where [HD].[Holiday-
Date]=@CurrentDate And [HD].[Country]=@Country)
            Begin
                Insert @DaysCount ( [CalendarDate] )
                Values  ( @CurrentDate )
            end
            Set @CurrentDate=DateAdd(Day,1,@CurrentDate)
        END


        Select  @DaysToExclude = Count(1)
        From    @DaysCount As [DC]

        Return @DaysToExclude;
    End;
GO
EXEC sp_addextendedproperty N'MS_Description', N'generates the number of working
days between two dates', 'SCHEMA', N'Process', 'FUNCTION', N'Udf_WorkingDays', NULL,
NULL
GO
```

Author: Johnson, Chris

Project> BORN> User databases> BlackBox> Programmability> Functions> Scalar-valued Functions> Process.Udf_WorkingDays

Uses

[Lookups].[HolidayDays]
[Process]

Used By

[Report].[UspResults_InventoryInspectionTimes]
[Report].[UspResults_SalesOrderKPI]

Author: Johnson, Chris