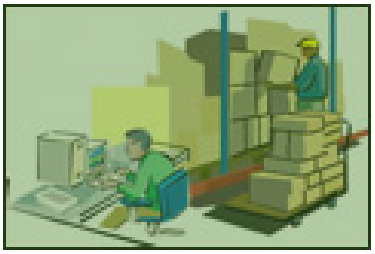
USAID | DELIVER PROJECT

Bangladesh: WIMS Technical Manual





WIMS Version 1.0

Warehouse Inventory Management System for DGFP

2010

This publication was produced for review by the U.S. Agency for International Development. It was prepared by the USAID | DELIVER PROJECT, Task Order 1.

Bangladesh: WIMS Technical Manual

USAID | DELIVER PROJECT, Task Order 1
The USAID | DELIVER PROJECT, Task Order 1, is funded by the U.S. Agency for International Development under contract no. GPO-I-01-06-00007-00, beginning September 29, 2006. Task Order 1 is implemented by John Snow, Inc., in collaboration with PATH; Crown Agents Consultancy, Inc.; Abt Associates; Fuel Logistics Group (Pty) Ltd.; UPS Supply Chain Solutions; The Manoff Group; and 3i Infotech. The project improves essential health commodity supply chains by strengthening logistics management information systems, streamlining distribution systems, identifying financial resources for procurement and supply chain operations, and enhancing forecasting and procurement planning. The project also encourages policymakers and donors to support logistics as a critical factor in the overall success of their health care mandates.

Recommended Citation
USAID | DELIVER PROJECT, Task Order 1. 2010. *Bangladesh: WIMS Technical Manual.* Arlington, Va.: USAID | DELIVER PROJECT, Task Order 1.

USAID | DELIVER PROJECT
John Snow, Inc.
1616 Fort Myer Drive, 11th Floor
Arlington, VA 22209 USA
Phone: 703-528-7474
Fax: 703-528-7480
Email: askdeliver@jsi.com
Internet: deliver.jsi.com

THE WIMS SOFTWARE **4**

SYSTEM REQUIREMENTS **4**

DATA FLOW DIAGRAM **4**

RECEIVE ITEMS FROM WAREHOUSE/SUPPLIER 4
ISSUE ITEMS TO FACILITY 4
GENERATE GATE PASS 4
GENERATE ADJUSTMENT INVOICE 4

..... **4**

PHYSICAL DATA MODEL **4**

TABLES **4**

A. PARAMETER TABLES 4
AdjType *4*
Designation *4*
Division *4*
District *4*
FGroup *4*
Facility *4*
Employee *4*
FormList *4*
FormAccess *4*
IndentDesig *4*
SQLLog *4*
SQLUpdate *4*
Supplier *4*
UnitOfMeas *4*
Warehouse *4*
MetaTable *4*
ItemGroup *4*
ItemList *4*
ItemListLot *4*
B. INFORMATION OR DETAILS TABLES 4
RecInv *4*
RecInvItems *4*
Indent *4*
IndentItems *4*
AdjInv *4*
AdjInvItems *4*
GatePassItems *4*
C. PROCESSED DATA TABLES 4
tmpStockDetails *4*

SCRIPT FOR DATABASE AND TABLE CREATION **4**

STORED PROCEDURES **4**

GetAdjInvManyInfo *4*
GetAdjInvOneInfo *4*
GetAdjInvTree *4*
GetAdjTypeList *4*
GetDesigList *4*
GetDistDetails *4*
GetDistTree *4*
GetDivisionList *4*
GetEmpAccess *4*

GetEmpList *4*
GetEmpListForOpt *4*
GetEmpListLogin *4*
GetEmpManyInfo *4*
GetEmpOneInfo *4*
GetEmpTree *4*
GetFGroupList *4*
GetFacilityDetails *4*
GetFacilityList *4*
GetFacilityListForOpt *4*
GetFacilityTree *4*
GetGPassManyInfo *4*
GetGPassOneInfo *4*
GetGPassTree *4*
GetGroupDetails *4*
GetGroupList *4*
GetIndentDesigList *4*
GetInvForGPass *4*
GetIssueInvManyInfo *4*
GetIssueInvOneInfo *4*
GetIssueInvTree *4*
GetItemCurStockQty *4*
GetItemDetailsNew *4*
GetItemListSelection *4*
GetItemLotDetails *4*
GetItemTransCount *4*
GetLotPickListForIssue *4*
GetMetaTableList *4*
GetRecInvManyInfo *4*
GetRecInvOneInfo *4*
GetRecInvTree *4*
GetStockSummaryLotwise *4*
GetSupplierDetails *4*
GetSupplierList *4*
GetUnitOfMeasList *4*
GetWarehouseOneInfo *4*
PostToStock_AdjInv *4*
PostToStock_IssueInv *4*
PostToStock_RecInv *4*

WIMS MENU STRUCTURE **4**

REPORTS **4**

AdjInv.rpt *4*
Districts.rpt *4*
FacilityList.rpt *4*
FacilityListByDist.rpt *4*
GatePass.rpt *4*
GatePassItems.rpt *4*
IndentInv.rpt, IssueInv.rpt *4*
ItemGroups.rpt *4*
RecInv.rpt *4*
RecInvFac.rpt *4*
StockDetails.rpt *4*
StockDetails_byFacility.rpt *4*
StockSummary.rpt *4*
StockSummaryAny.rpt *4*

StockSummaryLot.rpt *4*
StockSummaryLotAny.rpt *4*
Suppliers.rpt *4*

DATA ENTRY FORMS **4**

frmIssueInv form *4*

The WIMS software

WIMS (Warehouse Inventory Management System) is software for maintaining stock status of commodities in any warehouse. It's a fully menu-driven, client-server graphical user interface and easy to use. WIMS was developed using Visual Basic in the front end and SQL Server at the back end. WIMS has the following features:

- Issue, receive of commodities to and from different facilities
- Password-protected access control
- Issue of commodities using FEFO (first–expiry, first-out) method
- Adjustment due to loss, expiry, return, etc.

Family planning commodities are distributed through a three-tier distribution system. Central warehouse and Regional Warehouse (RWH) Chittagong receive commodities from external suppliers, which are then distributed to all of the other warehouses. The RWHs issue the commodities to Upazila Family Planning Office for distribution to filed staff. The main task of a warehouse, therefore, is to issue and receive commodities and maintain stock. When a Central Warehouse (CWH) or RWH receives commodities, they are added to stock by preparing a Receive Invoice. When an RWH issues commodities, it is done through Issue Invoice. In cases of commodity return, loss, damage, theft, or physical inventory, all are recorded through an Adjustment Invoice. At any time, users can get the current stock status and any previous stock status.

System Requirements

The following resources are recommended for use with WIMS:

Components	Specifications
CPU	Pentium III or higher
Memory	128 MB
Hard Drive	40 MB
Video Adapter	SVGA with 800 x 600 resolution
Operating System	Windows 2000, Windows XP
Database	SQL Server 2000

In a multiuser environment, SQL Server is required. WIMS does not require MS Office, but if a user needs to export a report in Word or Excel file format, then MS Office must be installed in the target machine.

The following software is required during development of WIMS:

Software	Specifications
Operating System	Windows 2000, Windows XP
Database	SQL Server 2000
Development Tools	Visual Basic 6
Reporting	Crystal Reports 8 or higher
Custom components	MyOCX.ocx, Subclass.ocx, OLEGUIDS.tlb, ISHF_Ex.tlb

To prepare a PC for development, users must follow the steps below:

1. Install SQL Server 2000, Visual Basic 6, then Crystal Reports.

2. Copy the VB project directory from CD-ROM to a hard disk. To run the project, you must first install the custom components.
3. Install MyOCX.ocx and Subclass.ocx using regsvr32 tool. Go to DOS prompt and type: regsvr32 <full path of ocx>.
4. Install the type libraries, OLEGUIDS.tlb, ISHF_Ex.tlb, using the third-party tool, VBRegTLB, available on CD-ROM. Start VBRegTLB, find the tlb files, and press Registered for every tlb not yet registered.
5. Now you are ready to open the VB project for editing.

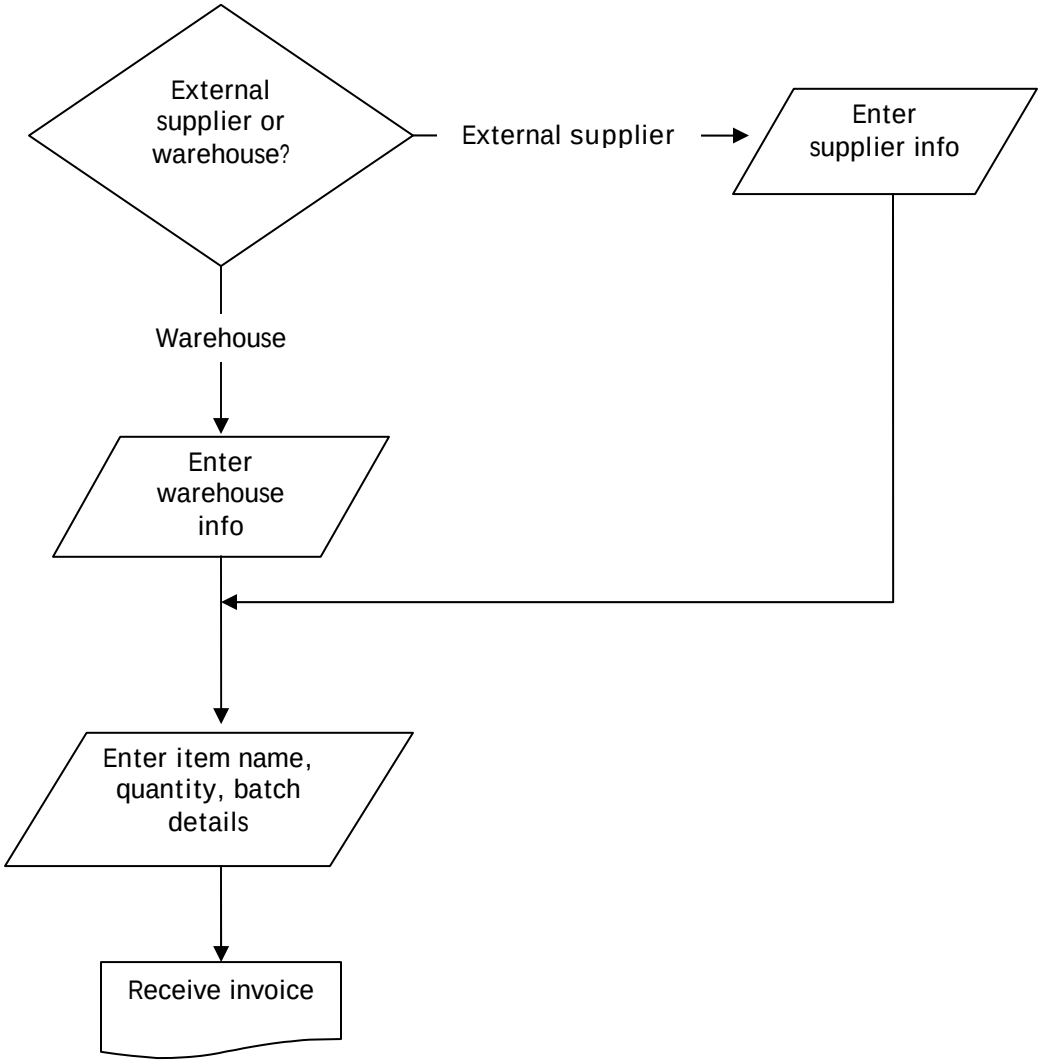
Naming Conventions Used

Following are the naming conventions used for different objects of the software:

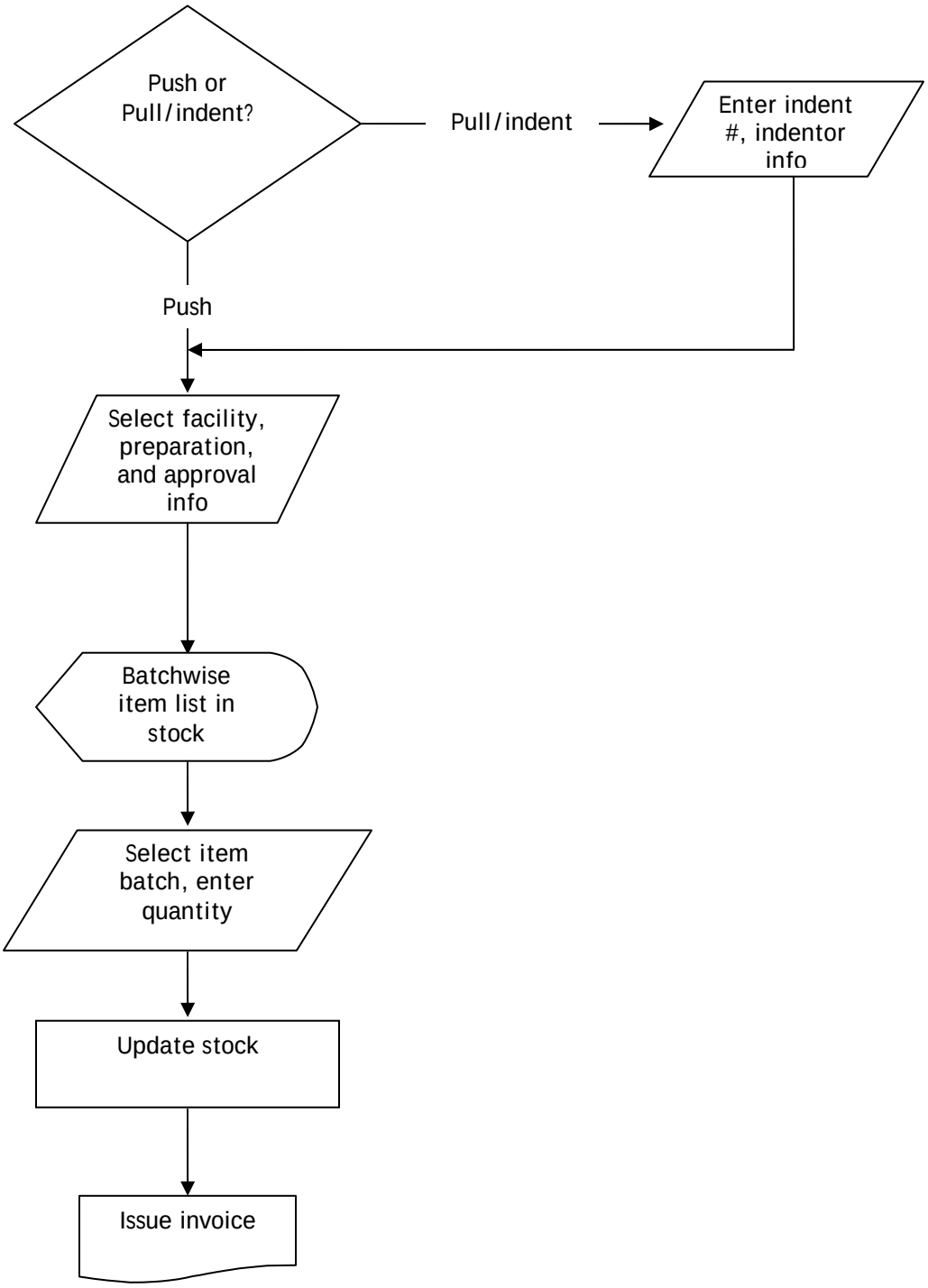
Component	Convention
▪ Database Table Names	▪ *None specific*
▪ Stored Procedure Names	▪ All SP returns data that start with **Get**; those used as report data source have **Rpt** at the end; those that update invoices start with **PostToStock**
▪ Data Entry Form Names	▪ Starts with **frm**
▪ Form Variable Names	▪ *None specific*
▪ Report Names	▪ *None specific*

Data Flow Diagram

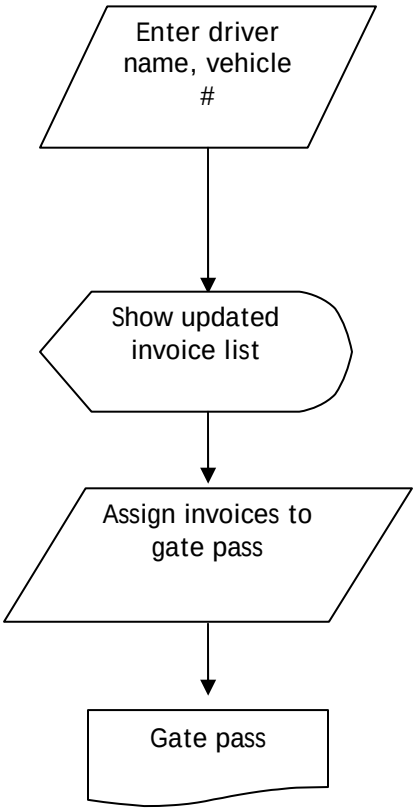
Receive Items from Warehouse/Supplier



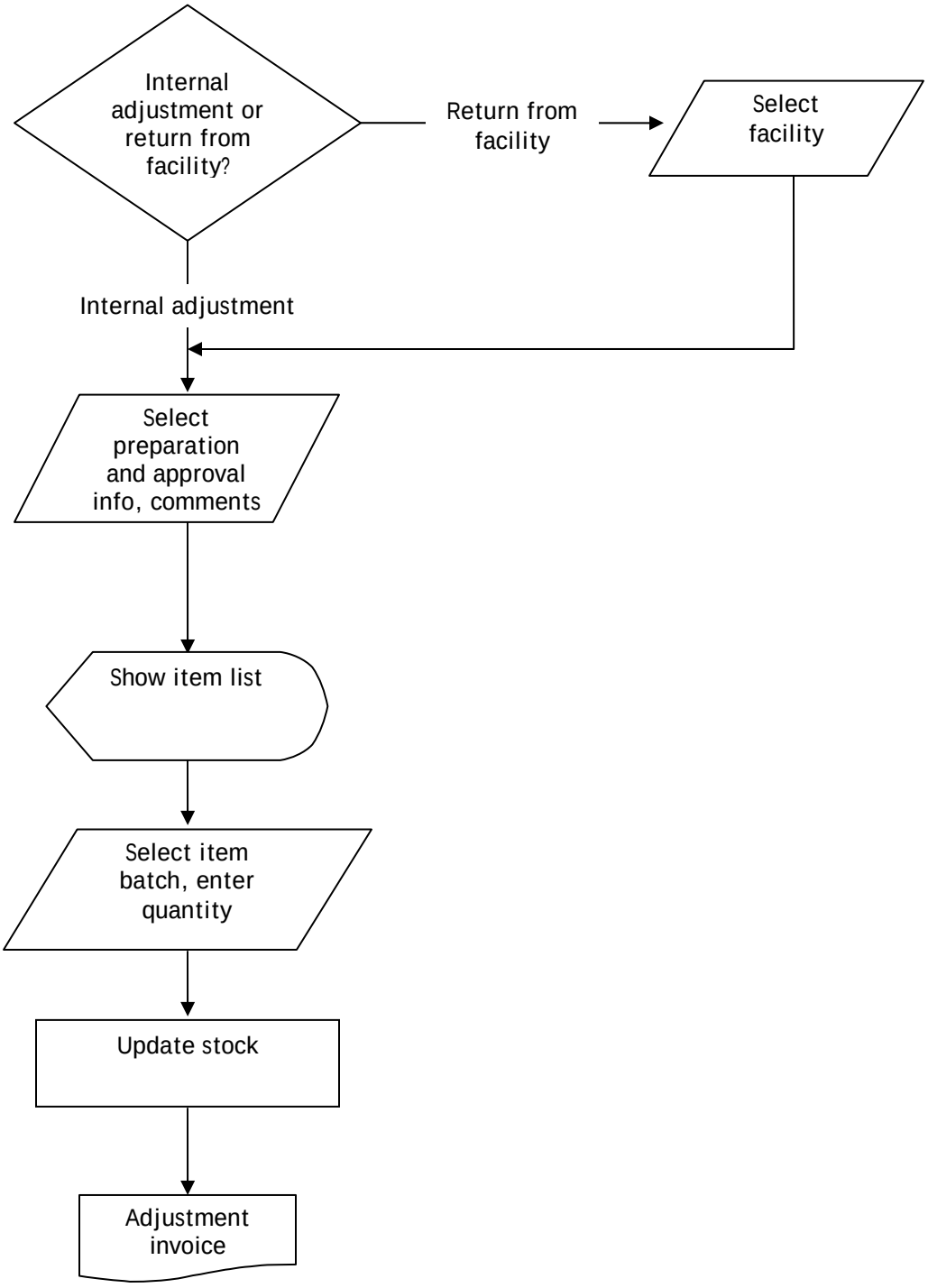
Issue Items to Facility



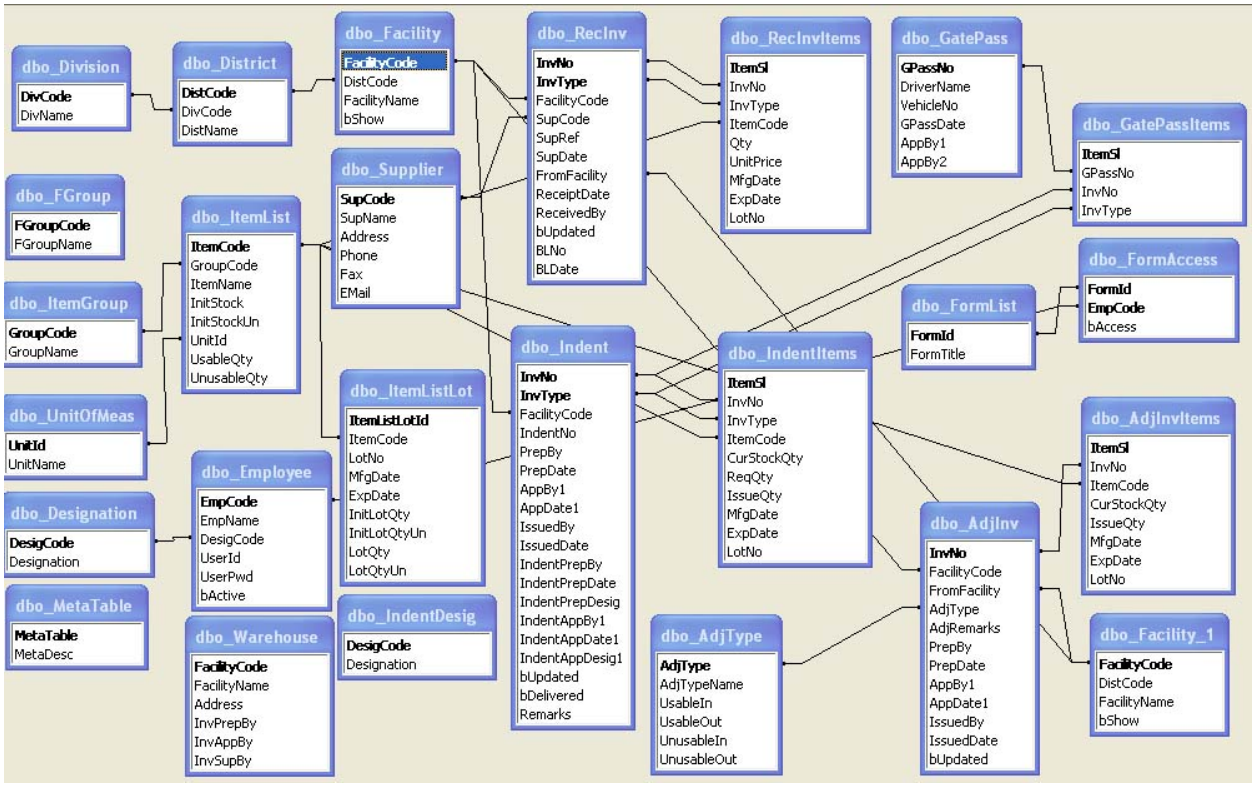
Generate Gate Pass



Generate Adjustment Invoice



Physical data Model



Tables

There are a total of 28 tables, which can be grouped into 3 categories:

A. Parameter Tables
B. Information or Details Tables
C. Processed Data Tables

A. Parameter Tables

These tables contain parameters, or lists of codes, that are used during data entry in Details Tables. Data in these tables are updated very infrequently. Following is the list of parameter Tables, along with their properties and description of fields.

Description of Legends:
P – Primary Key
F – Foreign Key
U – Unique Key
M – Mandatory (Not NULL)

AdjType

Adjustment invoices can be categorized into 6 types. These adjustment types are listed in AdjType table. When entering adjustment invoices, one of these adjustment types is selected from list. Each adjustment type has 4 associated flags. The flags hold 0 or 1 and are used during stock summarization.

Attribute	Field Name	Data Type	Size	P	F	U	M	Description
Adjustment Type	AdjType	smallint		Y		Y	Y	Numeric adjustment type incremented by 1
Adjustment Type Name	AdjTypeName	varchar	40			Y	Y	Descriptive name of adjustment type
4 Flags Used During Adjustment Calculations	UsableIn	smallint					Y	0 and 1 are set in the 4 flags to be used during calculations regarding adjustment
	UsableOut	smallint					Y	
	UnusableIn	smallint					Y	
	UnusableOut	smallint					Y	

Designation

This table contains warehouse employee designations. When entering employee information, one designation is selected from the list for a specific employee.

Attribute	Field Name	Data Type	Size	P	F	U	M	Description
Designation Code	DesigCode	smallint		Y		Y	Y	Unique code for each type of designation incremented by 1
Designation Name	Designation	varchar	50			Y	Y	Descriptive designation name

Division

This table lists the 6 divisions in Bangladesh. When new divisions are added, they can be included in this table.

Attribute	Field Name	Data Type	Size	P	F	U	M	Description
Division Code	DivCode	Smallint		Y		Y	Y	Unique code for each division incremented by 1
Division Name	DivName	Varchar	20			Y	Y	Name of division

District

This table lists the 64 districts of Bangladesh. When new districts are added, they can be included in this table.

Attribute	Field Name	Data Type	Size	P	F	U	M	Description
District Code	DistCode	Int		Y		Y	Y	Unique district code for each district incremented by 1
Division Code	DivCode	smallint			Y		Y	Foreign key of Division table: division code under which the district belongs
District Name	DistName	varchar	50			Y	Y	Name of district

FGroup

Facilities are arranged in a number of groups, each of which is identified by a 1-character Group Code. When entering a Facility, a group is selected from list.

Attribute	Field Name	Data Type	Size	P	F	U	M	Description
Facility Group Code	FGroupCode	Char	1	Y		Y	Y	Unique code for each type of facility
Facility Group Name	FGroupName	varchar	30				Y	Descriptive name of each type of facility code

Facility

The warehouses, thanas, and other organizations where commodities can be issued and received are listed in this table.

Attribute	Field Name	Data Type	Size	P	F	U	M	Description
Facility Code	FacilityCode	varchar	4	Y		Y	Y	Unique facility code for each facility: first character, facility group, and last 3 characters, sequential number
District Code	DistCode	int			Y		Y	Foreign key to Districts table: district code under which the facility belongs
Facility Name	FacilityName	varchar	100			Y	Y	Descriptive name of facility
Show	bShow	bit					Y	Whether to show it during invoice entry

Employee

Each warehouse employs a number of employees who use WIMS. Information about the employees, as well as their User IDs and Passwords to log into the system, are stored in this table.

Attribute	Field Name	Data Type	Size	P	F	U	M	Description
Employee Code	EmpCode	Int		Y		Y	Y	Unique employee code for each employee incremented by 1
Employee Name	EmpName	varchar	100				Y	Name of employee
Designation Code	DesigCode	smallint			Y		Y	Foreign key to Designation table: designation code of employee
User ID	UserId	varchar	8			Y	Y	Unique User ID for each employee used during login
User Password	UserPwd	varchar	8				Y	Password for each employee used during login
Active	bActive	bit					Y	Whether employee is currently working

FormList

This table lists the data entry forms in the software.

Attribute	Field Name	Data Type	Size	P	F	U	M	Description
Form ID	FormId	varchar	20	Y		Y	Y	Unique name for each data entry form
Form Title	FormTitle	varchar	50				Y	Descriptive name of each data entry form

FormAccess

Employee access to data entry forms is listed in this table. There are two types of access—View and Edit. With View access, users get Read Only access to the forms; they cannot edit anything. With Edit access, users can enter new records, edit old ones, delete records, etc.

Attribute	Field Name	Data Type	Size	P	F	U	M	Description
Form ID	FormId	varchar	20	Y	Y	Y	Y	Foreign key to FormList: form for which access rights are to be given
Employee Code	EmpCode	int		Y	Y		Y	Foreign key to employees: employee code for which permission to access a form will be given
Access	bAccess	smallint					Y	Permission to View or Edit the form

IndentDesig

This table lists the designations of persons who prepare/approve indent vouchers from the receiving facility.

Attribute	Field Name	Data Type	Size	P	F	U	M	Description
Designation Code	DesigCode	smallint		Y		Y	Y	Unique designation code incremented by 1
Designation Name	Designation	Varchar	50			Y	Y	Designation of persons who prepare/approve indent vouchers

SQLLog

This table acts as an Audit Trail for the software. Any inserts, updates, or delete commands issued against the database are listed in this table and can be viewed later for tracking purposes. Also, if there are multiple databases, then these changes can be transported to another database and executed to make the databases identical.

Attribute	Field Name	Data Type	Size	P	F	U	M	Description
Log Sequence	LogSeq	int		Y		Y	Y	Sequential number incremented by 1
User Name	UserName	varchar	30				Y	Holds the logged-on User ID that initiated the SQL query
Terminal	Terminal	varchar	30				Y	From which terminal the query was initiated
Time Stamp	TStamp	datetime					Y	Specific time when the query was initiated
SQL Text	SqlText	varchar	1000				Y	SQL text of the query
Copied	Copied	bit					Y	Whether the update was copied to another server

SQLUpdate

This table is not used currently; updates generated in SQLLog table can be transferred to another database. In the new database, the records are entered into SQLUpdate table and can be updated later in the database.

Attribute	Field Name	Data Type	Size	P	F	U	M	Description
Log Sequence	LogSeq	int		Y		Y	Y	Sequential number incremented by 1
User Name	UserName	varchar	8				Y	Holds the logged-on User ID that initiated the SQL query
Terminal	Terminal	varchar	30				Y	From which terminal the query was initiated
Time Stamp	TStamp	datetime					Y	Specific time when the query was initiated
SQL Text	SqlText	varchar	1000				Y	SQL text of the query
Updated	Updated	bit					Y	Whether the query was updated in the destination server

Supplier

The list of external suppliers is stored in this table. External suppliers refers to suppliers other than warehouses or facilities in the Directorate General of Family Planning (DGFP) distribution system.

Attribute	Field Name	Data Type	Size	P	F	U	M	Description
Supplier Code	SupCode	int		Y		Y	Y	Unique code for each supplier incremented by 1
Supplier Name	SupName	varchar	50			Y	Y	Name of supplier
Address	Address	varchar	150					Full address of supplier
Phone	Phone	varchar	100					Phone numbers of supplier
Fax	Fax	varchar	100					Fax number of supplier
Email	EMail	varchar	100					Email address of supplier

UnitOfMeas

This table stores units of measurement for commodities. One unit is selected for each item; new items are entered in the ItemList table.

Attribute	Field Name	Data Type	Size	P	F	U	M	Description
Unit ID	UnitId	smallint		Y		Y	Y	Unique ID for each unit incremented by 1
Unit Name	UnitName	varchar	20			Y	Y	Name of unit

Warehouse

This table is filled before data entry starts. The first 3 fields contain information about the facility where the software will be used. In last 3 fields you select the employee codes of the persons who will be involved in the invoice-preparation process. These data will be used to fill in data entry fields automatically when invoices are prepared.

Attribute	Field Name	Data Type	Size	P	F	U	M	Description
Facility Code	FacilityCode	varchar	4	Y		Y	Y	Foreign key to Facility table: select own code of the facility
Facility Name	FacilityName	varchar	100				Y	Name of own facility
Address of Facility	Address	varchar	100				Y	Address of own facility
Invoice Prepared By	InvPrepBy	int						Code of employee who generally prepares the invoice
Invoice Approved By	InvAppBy	int						Code of employee who generally approves the invoice
Supplied By	InvSupBy	int						Code of employee who generally supplies goods

MetaTable

There is one generic data entry form for data entry in the basic lookup tables. The entry form has a list filled with table names; after you select the table name, the records of that table will appear in the grid. This table contains those table names.

Attribute	Field Name	Data Type	Size	P	F	U	M	Description
Meta Table Name	MetaTable	varchar	100	Y		Y	Y	Name of database table
Meta Table Description	MetaDesc	varchar	100			Y		Description of meta table

ItemGroup

Commodities in the software are categorized into several groups; the names of those groups are listed in this table.

Attribute	Field Name	Data Type	Size	P	F	U	M	Description
Group Code	GroupCode	char	3	Y		Y	Y	3-digit unique code for each item group
Group Name	GroupName	varchar	100			Y	Y	Descriptive name of each group

ItemList

A list of items are saved in this table. Initially when the software starts, you put the item names together with their initial quantity and unit of measurement. Whenever a new item enters the system, you have to put the item and its unit in this table.

Attribute	Field Name	Data Type	Size	P	F	U	M	Description
Item Code	ItemCode	varchar	10	Y		Y	Y	Unique code for each item: 3-character group code and 3-digit sequential unique number
Group Code	GroupCode	Char	3		Y		Y	Foreign key of ItemGroup table: item belongs to this group
Item Name	ItemName	varchar	150				Y	Name of item
Initial Stock	InitStock	decimal	18,0				Y	Initial stock of item
Initial Unusable Stock	InitStockUn	decimal	18,0				Y	Initial unusable stock of item
Unit ID	UnitId	smallint			Y		Y	Foreign key to UnitOfMeas table: unit of measurement of the item
Total Usable Quantity	UsableQty	decimal	18,0				Y	Current total usable stock quantity of the item
Total Unusable Quantity	UnusableQty	decimal	18,0				Y	Current total unusable stock quantity of the item

ItemListLot

This table contains item lot or batch information. An item may have one or more lots, and each lot can be identified by 3 fields—lot number, manufacturing date, and expiry date.

Attribute	Field Name	Data Type	Size	P	F	U	M	Description
Item List Lot ID	ItemListLotId	Int		Y		Y	Y	Sequential lot serial
Item Code	ItemCode	varchar	10		Y		Y	Foreign key to ItemList: code name of item
Lot No	LotNo	varchar	100					Lot number of specific item
Manufacturing Date	MfgDate	datetime						Manufacturing date of the lot
Expiry Date	ExpDate	datetime						Expiry date of the lot
Initial Lotwise Quantity	InitLotQty	decimal	18,0					Initial quantity for this lot
Initial Unusable Lotwise Quantity	InitLotQtyUn	decimal	18,0					Initial unusable quantity for this lot
Total Current Lot Quantity	LotQty	decimal	18,0					Total current usable lot quantity
Total Current Unusable Lot Quantity	LotQtyUn	decimal	18,0					Total current unusable lot quantity

B. Information or Details Tables

RecInv

This is the master table for storing receive invoice. There are two types of sources from which to receive—one is an external source and the other is an internal transfer between warehouses.

Attribute	Field Name	Data Type	Size	P	F	U	M	Description
Invoice No	InvNo	int		Y		Y	Y	Sequential invoice number incremented by 1
Invoice Type	InvType	smallint		Y		Y	Y	Two types of sources from which to receive—external supplier and warehouse
Facility Code	FacilityCode	varchar	4		Y		Y	Foreign key to Facility table: code of facility that has received the commodities
Supplier Code	SupCode	int			Y			Foreign key to Supplier table: code of supplier that supplied the commodities
Supplier Challan	SupRef	varchar	30				Y	Challan number from supplier's delivery invoice
Supplier Date	SupDate	datetime					Y	Date of supply
From Facility	FromFacility	varchar	4		Y			Foreign key to Facility table: if received from warehouse, then facility that sent the commodities
Receipt Date	ReceiptDate	datetime					Y	Date of receipt of invoice
Updated	bUpdated	bit					Y	Whether the invoice is updated to stock
Bill of Lading No	BLNo	varchar	200					If external supply, then bill of lading number
Bill of Lading Date	BLDate	datetime						If external supply, then bill of lading date

RecInvItems

Information about the items received, their quantity, and batchwise information are stored in this table. There can be multiple records in this table for a single invoice.

Attribute	Field Name	Data Type	Size	P	F	U	M	Description
Item Serial	ItemSl	int		Y		Y	Y	Unique serial number
Invoice No	InvNo	int			Y		Y	Foreign key to RecInv table: receive voucher number and type
Invoice Type	InvType	smallint			Y		Y	
Item Code	ItemCode	varchar	10		Y		Y	Foreign key to ItemList table: code of item that was received
Received Quantity	Qty	decimal	18,0					Quantity received
Unit Price	UnitPrice	decimal	18,0					Unit price*******
Manufacturing Date	MfgDate	datetime						Manufacturing date of the lot
Expiry Date	ExpDate	datetime						Expiry date of the lot
Lot No	LotNo	varchar	100					Lot number, if any

Indent

This table stores information regarding issuing of commodities to different stores. There is one record in this table for each issue invoice. Issue can be of two types—push issue based on allotment, and pull issue based on indent from receiving store.

Attribute	Field Name	Data Type	Size	P	F	U	M	Description
Invoice No	InvNo	Int		Y		Y	Y	Sequential invoice number for each issue invoice
Invoice Type	InvType	smallint		Y		Y	Y	Two types of invoice—push and pull (indent)
Facility Code	FacilityCode	varchar	4		Y		Y	Foreign key of Facility table: facility for which commodities are issued
Indent No	IndentNo	varchar	50					If issue is based on indent, receiving facility's indent number
Prepared By	PrepBy	Int					Y	Code of employee who prepared the invoice
Preparation Date	PrepDate	datetime					Y	Date of invoice was prepared
Approved By	AppBy1	Int					Y	Code of employee who approved the invoice
Approved By Date	AppDate1	datetime					Y	Date of approval
Issued By	IssuedBy	Int					Y	Code of employee who issued the commodities
Issue Date	IssuedDate	datetime					Y	Date of issue
Indent Prepared By	IndentPrepBy	varchar	100					Name of person in receiving facility who prepared the indent voucher
Indent Preparation Date	IndentPrepDate	datetime						Date of indent preparation
Indent Preparation Designation	IndentPrepDesig	smallint						Designation of person who prepared the invoice
Indent Approved By	IndentAppBy1	varchar	100					Name of person who approved the indent
Indent Approval Date	IndentAppDate1	datetime						Date ident was approved
Indent Approval Designation	IndentAppDesig1	smallint						Designation of person who approved the invoice
Updated	bUpdated	Bit					Y	Whether the invoice is updated to stock
Delivered	bDelivered	Bit					Y	Whether the commodities in the invoice are delivered through gate pass
Remarks	Remarks	varchar	200					Any comments

IndentItems

Every issue voucher has one or more records in this table for each of the items issued. Every record has item name, batch information, and quantity issued.

Attribute	Field Name	Data Type	Size	P	F	U	M	Description
Item Serial	ItemSl	int		Y		Y	Y	Sequential serial number
Invoice No	InvNo	Int			Y		Y	Foreign key to Indent table: invoice number and type
Invoice Type	InvType	smallint			Y		Y	
Item Code	ItemCode	varchar	10		Y		Y	Foreign key to ItemList: code of item being issued

Current Stock Quantity	CurStockQty	decimal	18,0					Current stock quantity of specific item
Requisition Quantity	ReqQty	decimal	18,0					Quantity the indentor requested
Issued Quantity	IssueQty	decimal	18,0					Actual quantity issued
Manufacturing Date	MfgDate	datetime						Specific lot information
Expiry Date	ExpDate	datetime						
Lot No	LotNo	varchar	100					

AdjInv

Adjustment invoice information stored in this table. Every adjustment invoice has an adjustment type associated.

Attribute	Field Name	Data Type	Size	P	F	U	M	Description
Invoice No	InvNo	int		Y		Y	Y	Unique adjustment invoice number incremented by 1
Facility Code	FacilityCode	varchar	4		Y		Y	Foreign key of Facility table: facility for which adjustment is performed
From Facility	FromFacility	varchar	4		Y		Y	Foreign key of Facility table: in case of return, which facility is returning the commodity
Adjustment Type	AdjType	smallint			Y		Y	Foreign key of AdjType table: what type of adjustment is performed
Remarks	AdjRemarks	varchar	200					Comments regarding the adjustment
Invoice Prepared By	PrepBy	int					Y	Code of employee who prepared the invoice
Date of Invoice Preparation	PrepDate	datetime					Y	Invoice preparation date
Invoice Approved By	AppBy1	int					Y	Code of employee who approved the invoice
Date of Approval	AppDate1	datetime					Y	Invoice approval date
Commodities Issued By	IssuedBy	int						Code of employee who issued the commodities
Date of Issue	IssuedDate	datetime						Invoice issue date
Update Flag	bUpdated	bit					Y	Whether the invoice is updated to stock

AdjInvItems

Item details information of an adjustment invoice is stored in this table. Every item adjusted contains one record.

Attribute	Field Name	Data Type	Size	P	F	U	M	Description
Item Serial	ItemSl	int		Y		Y	Y	Sequential number for each adjustment item entry
Invoice No	InvNo	int			Y		Y	Foreign key of AdjInv table: adjustment invoice number
Item Code	ItemCode	varchar	10		Y		Y	Foreign key of ItemList table: adjustment for this item code
Current Stock Quantity	CurStockQty	decimal	18,0					Current balance of specific item in stock
Adjusted Quantity	IssueQty	decimal						Quantity of item adjusted
Manufacturing Date	MfgDate	datetime						Manufacturing date of the item lot
Expiry Date	ExpDate	datetime						Expiry date of the item lot
Lot No	LotNo	varchar	100					Lot number of the item lot

GatePass

Information about gate pass is stored in this table.

Attribute	Field Name	Data Type	Size	P	F	U	M	Description
Gate Pass No	GPassNo	int		Y		Y	Y	Unique Gate Pass no. incremented by 1
Driver Name	DriverName	varchar	100					Name of driver driving the vehicle
Vehicle No	VehicleNo	varchar	100					Number of vehicles carrying the goods
Gate Pass Date	GPassDate	datetime					Y	Date of Gate Pass preparation
Approved By 1	AppBy1	int						Employee 1 who approved the invoice
Approved By 2	AppBy2	int						Employee 2 who approved the invoice

GatePassItems

Every gate pass links a number of invoices, which are stored in this table.

Attribute	Field Name	Data Type	Size	P	F	U	M	Description
Item Serial	ItemSl	Int		Y		Y	Y	Unique item serial number
Gate Pass No	GPassNo	Int			Y		Y	Foreign key to Gate Pass table: gate pass number
Invoice No	InvNo	Int			Y		Y	Foreign key to Indent table: Invoice Type and Number that are attached to this Gate Pass
Invoice Type	InvType	smallint			Y		Y	

C. Processed Data Tables

tmpStockDetails

This table contains pre-processed data, which are used to produce summary reports. Every set of records is identified by a unique code. Records older than one day are deleted before a report is generated.

Attribute	Field Name	Data Type	Size	P	F	U	M	Description
Unique Code	UCode	char	32				Y	Unique code to identify a group of records
Current Date	CurDate	datetime					Y	Date of report printing
Group Code	GroupCode	Char	3				Y	Item Group Code
Group Name	GroupName	varchar	100				Y	Item Group Name
Item Code	ItemCode	varchar	10				Y	Item Code
Item Name	ItemName	varchar	150				Y	Name of item
Supplier Name	Supplier	varchar	100					Name of supplier (external/warehouse)
Receiver Name	Receiver	varchar	100					Name of receiving warehouse
Invoice No	InvoiceNo	varchar	100					Invoice number (receive/indent/issue/adjustment)
Stock Date	StockDate	datetime						Date of stock
Total Usable Quantity In	TotQtyIn	decimal	18,0					Total quantity of item that was added to stock (initial stock + receive qty + adjusted qty)
Total Usable Quantity Out	TotQtyOut	decimal	18,0					Total quantity of item that was removed from stock (initial stock-issued, qty-adjusted qty)
Total Unusable Quantity In	TotUnQtyIn	decimal	18,0					Total unusable quantity of item that was added to stock (initial stock + receive qty + adjusted qty)
Total Unusable Quantity Out	TotUnQtyOut	decimal	18,0					Total unusable quantity of item that was removed from stock (initial stock-issued qty-adjusted qty)
Unit Name	UnitName	varchar	20					Name of item unit
Invoice Type	InvType	varchar	100					A description of type of invoice

Script for Database and Table Creation

```
IF EXISTS (SELECT name FROM master.dbo.sysdatabases WHERE name = N'wims')
        DROP DATABASE [wims]
GO

CREATE DATABASE [wims]  ON (NAME = N'wims_Data', FILENAME = N'C:\Program Files\Microsoft SQL
Server\MSSQL\data\wims_Data.MDF', SIZE = 20, FILEGROWTH = 10%) LOG ON (NAME = N'wims_Log', FILENAME
= N'C:\Program Files\Microsoft SQL Server\MSSQL\data\wims_Log.LDF', SIZE = 2, FILEGROWTH = 10%)
 COLLATE SQL_Latin1_General_CP1_CI_AS
GO

exec sp_dboption N'wims', N'autoclose', N'true'
GO

exec sp_dboption N'wims', N'bulkcopy', N'false'
GO

exec sp_dboption N'wims', N'trunc. log', N'true'
GO

exec sp_dboption N'wims', N'torn page detection', N'true'
GO

exec sp_dboption N'wims', N'read only', N'false'
GO

exec sp_dboption N'wims', N'dbo use', N'false'
GO

exec sp_dboption N'wims', N'single', N'false'
GO

exec sp_dboption N'wims', N'autoshrink', N'true'
GO

exec sp_dboption N'wims', N'ANSI null default', N'false'
GO

exec sp_dboption N'wims', N'recursive triggers', N'false'
GO

exec sp_dboption N'wims', N'ANSI nulls', N'false'
GO

exec sp_dboption N'wims', N'concat null yields null', N'false'
GO

exec sp_dboption N'wims', N'cursor close on commit', N'false'
GO

exec sp_dboption N'wims', N'default to local cursor', N'false'
GO

exec sp_dboption N'wims', N'quoted identifier', N'false'
GO

exec sp_dboption N'wims', N'ANSI warnings', N'false'
GO

exec sp_dboption N'wims', N'auto create statistics', N'true'
GO

exec sp_dboption N'wims', N'auto update statistics', N'true'
GO
```

```
use [wims]
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[FK_AdjInvItems_AdjInv]') and
OBJECTPROPERTY(id, N'IsForeignKey') = 1)
ALTER TABLE [dbo].[AdjInvItems] DROP CONSTRAINT FK_AdjInvItems_AdjInv
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[FK_AdjInv_AdjType]') and
OBJECTPROPERTY(id, N'IsForeignKey') = 1)
ALTER TABLE [dbo].[AdjInv] DROP CONSTRAINT FK_AdjInv_AdjType
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[FK_Employee_Designation]') and
OBJECTPROPERTY(id, N'IsForeignKey') = 1)
ALTER TABLE [dbo].[Employee] DROP CONSTRAINT FK_Employee_Designation
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[FK_Facility_District]') and
OBJECTPROPERTY(id, N'IsForeignKey') = 1)
ALTER TABLE [dbo].[Facility] DROP CONSTRAINT FK_Facility_District
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[FK_District_Division]') and
OBJECTPROPERTY(id, N'IsForeignKey') = 1)
ALTER TABLE [dbo].[District] DROP CONSTRAINT FK_District_Division
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[FK_FormAccess_Employee]') and
OBJECTPROPERTY(id, N'IsForeignKey') = 1)
ALTER TABLE [dbo].[FormAccess] DROP CONSTRAINT FK_FormAccess_Employee
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[FK_AdjInv_Facility]') and
OBJECTPROPERTY(id, N'IsForeignKey') = 1)
ALTER TABLE [dbo].[AdjInv] DROP CONSTRAINT FK_AdjInv_Facility
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[FK_AdjInv_Facility1]') and
OBJECTPROPERTY(id, N'IsForeignKey') = 1)
ALTER TABLE [dbo].[AdjInv] DROP CONSTRAINT FK_AdjInv_Facility1
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[FK_Indent_Facility]') and
OBJECTPROPERTY(id, N'IsForeignKey') = 1)
ALTER TABLE [dbo].[Indent] DROP CONSTRAINT FK_Indent_Facility
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[FK_RecInv_Facility]') and
OBJECTPROPERTY(id, N'IsForeignKey') = 1)
ALTER TABLE [dbo].[RecInv] DROP CONSTRAINT FK_RecInv_Facility
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[FK_RecInv_Facility1]') and
OBJECTPROPERTY(id, N'IsForeignKey') = 1)
ALTER TABLE [dbo].[RecInv] DROP CONSTRAINT FK_RecInv_Facility1
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[FK_FormAccess_FormList]') and
OBJECTPROPERTY(id, N'IsForeignKey') = 1)
ALTER TABLE [dbo].[FormAccess] DROP CONSTRAINT FK_FormAccess_FormList
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[FK_GatePassItems_GatePass]') and
OBJECTPROPERTY(id, N'IsForeignKey') = 1)
ALTER TABLE [dbo].[GatePassItems] DROP CONSTRAINT FK_GatePassItems_GatePass
GO
```

```
if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[FK_GatePassItems_Indent]') and
OBJECTPROPERTY(id, N'IsForeignKey') = 1)
ALTER TABLE [dbo].[GatePassItems] DROP CONSTRAINT FK_GatePassItems_Indent
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[FK_IndentItems_Indent]') and
OBJECTPROPERTY(id, N'IsForeignKey') = 1)
ALTER TABLE [dbo].[IndentItems] DROP CONSTRAINT FK_IndentItems_Indent
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[FK_ItemList_ItemGroup]') and
OBJECTPROPERTY(id, N'IsForeignKey') = 1)
ALTER TABLE [dbo].[ItemList] DROP CONSTRAINT FK_ItemList_ItemGroup
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[FK_ItemListLot_ItemList]') and
OBJECTPROPERTY(id, N'IsForeignKey') = 1)
ALTER TABLE [dbo].[ItemListLot] DROP CONSTRAINT FK_ItemListLot_ItemList
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[FK_RecInvItems_RecInv]') and
OBJECTPROPERTY(id, N'IsForeignKey') = 1)
ALTER TABLE [dbo].[RecInvItems] DROP CONSTRAINT FK_RecInvItems_RecInv
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[FK_RecInv_Supplier]') and
OBJECTPROPERTY(id, N'IsForeignKey') = 1)
ALTER TABLE [dbo].[RecInv] DROP CONSTRAINT FK_RecInv_Supplier
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[FK_ItemList_UnitOfMeas]') and
OBJECTPROPERTY(id, N'IsForeignKey') = 1)
ALTER TABLE [dbo].[ItemList] DROP CONSTRAINT FK_ItemList_UnitOfMeas
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[Insert_ItemListLot]') and
OBJECTPROPERTY(id, N'IsTrigger') = 1)
drop trigger [dbo].[Insert_ItemListLot]
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[GetAdjInvManyInfo]') and
OBJECTPROPERTY(id, N'IsProcedure') = 1)
drop procedure [dbo].[GetAdjInvManyInfo]
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[GetAdjInvOneInfo]') and
OBJECTPROPERTY(id, N'IsProcedure') = 1)
drop procedure [dbo].[GetAdjInvOneInfo]
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[GetAdjInvRpt]') and OBJECTPROPERTY(id,
N'IsProcedure') = 1)
drop procedure [dbo].[GetAdjInvRpt]
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[GetAdjInvTree]') and OBJECTPROPERTY(id,
N'IsProcedure') = 1)
drop procedure [dbo].[GetAdjInvTree]
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[GetAdjTypeList]') and OBJECTPROPERTY(id,
N'IsProcedure') = 1)
drop procedure [dbo].[GetAdjTypeList]
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[GetDesigList]') and OBJECTPROPERTY(id,
N'IsProcedure') = 1)
drop procedure [dbo].[GetDesigList]
GO
```

```
if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[GetDistDetails]') and OBJECTPROPERTY(id,
N'IsProcedure') = 1)
drop procedure [dbo].[GetDistDetails]
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[GetDistTree]') and OBJECTPROPERTY(id,
N'IsProcedure') = 1)
drop procedure [dbo].[GetDistTree]
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[GetDivisionList]') and OBJECTPROPERTY(id,
N'IsProcedure') = 1)
drop procedure [dbo].[GetDivisionList]
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[GetEmpAccess]') and OBJECTPROPERTY(id,
N'IsProcedure') = 1)
drop procedure [dbo].[GetEmpAccess]
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[GetEmpList]') and OBJECTPROPERTY(id,
N'IsProcedure') = 1)
drop procedure [dbo].[GetEmpList]
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[GetEmpListForOpt]') and
OBJECTPROPERTY(id, N'IsProcedure') = 1)
drop procedure [dbo].[GetEmpListForOpt]
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[GetEmpListLogin]') and OBJECTPROPERTY(id,
N'IsProcedure') = 1)
drop procedure [dbo].[GetEmpListLogin]
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[GetEmpManyInfo]') and OBJECTPROPERTY(id,
N'IsProcedure') = 1)
drop procedure [dbo].[GetEmpManyInfo]
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[GetEmpOneInfo]') and OBJECTPROPERTY(id,
N'IsProcedure') = 1)
drop procedure [dbo].[GetEmpOneInfo]
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[GetEmpTree]') and OBJECTPROPERTY(id,
N'IsProcedure') = 1)
drop procedure [dbo].[GetEmpTree]
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[GetFGroupList]') and OBJECTPROPERTY(id,
N'IsProcedure') = 1)
drop procedure [dbo].[GetFGroupList]
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[GetFacilityDetails]') and OBJECTPROPERTY(id,
N'IsProcedure') = 1)
drop procedure [dbo].[GetFacilityDetails]
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[GetFacilityList]') and OBJECTPROPERTY(id,
N'IsProcedure') = 1)
drop procedure [dbo].[GetFacilityList]
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[GetFacilityListForOpt]') and
OBJECTPROPERTY(id, N'IsProcedure') = 1)
drop procedure [dbo].[GetFacilityListForOpt]
```

```
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[GetFacilityListRpt]') and OBJECTPROPERTY(id,
N'IsProcedure') = 1)
drop procedure [dbo].[GetFacilityListRpt]
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[GetFacilityTree]') and OBJECTPROPERTY(id,
N'IsProcedure') = 1)
drop procedure [dbo].[GetFacilityTree]
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[GetGPassManyInfo]') and
OBJECTPROPERTY(id, N'IsProcedure') = 1)
drop procedure [dbo].[GetGPassManyInfo]
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[GetGPassOneInfo]') and OBJECTPROPERTY(id,
N'IsProcedure') = 1)
drop procedure [dbo].[GetGPassOneInfo]
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[GetGPassTree]') and OBJECTPROPERTY(id,
N'IsProcedure') = 1)
drop procedure [dbo].[GetGPassTree]
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[GetGatePassItemsRpt]') and
OBJECTPROPERTY(id, N'IsProcedure') = 1)
drop procedure [dbo].[GetGatePassItemsRpt]
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[GetGatePassRpt]') and OBJECTPROPERTY(id,
N'IsProcedure') = 1)
drop procedure [dbo].[GetGatePassRpt]
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[GetGroupDetails]') and OBJECTPROPERTY(id,
N'IsProcedure') = 1)
drop procedure [dbo].[GetGroupDetails]
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[GetGroupList]') and OBJECTPROPERTY(id,
N'IsProcedure') = 1)
drop procedure [dbo].[GetGroupList]
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[GetIndentDesigList]') and
OBJECTPROPERTY(id, N'IsProcedure') = 1)
drop procedure [dbo].[GetIndentDesigList]
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[GetInvForGPass]') and OBJECTPROPERTY(id,
N'IsProcedure') = 1)
drop procedure [dbo].[GetInvForGPass]
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[GetIssueInvManyInfo]') and
OBJECTPROPERTY(id, N'IsProcedure') = 1)
drop procedure [dbo].[GetIssueInvManyInfo]
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[GetIssueInvOneInfo]') and
OBJECTPROPERTY(id, N'IsProcedure') = 1)
drop procedure [dbo].[GetIssueInvOneInfo]
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[GetIssueInvRpt]') and OBJECTPROPERTY(id,
N'IsProcedure') = 1)
```

```
drop procedure [dbo].[GetIssueInvRpt]
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[GetIssueInvTree]') and OBJECTPROPERTY(id,
N'IsProcedure') = 1)
drop procedure [dbo].[GetIssueInvTree]
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[GetItemCurStockQty]') and
OBJECTPROPERTY(id, N'IsProcedure') = 1)
drop procedure [dbo].[GetItemCurStockQty]
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[GetItemDetailsNew]') and
OBJECTPROPERTY(id, N'IsProcedure') = 1)
drop procedure [dbo].[GetItemDetailsNew]
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[GetItemListSelection]') and
OBJECTPROPERTY(id, N'IsProcedure') = 1)
drop procedure [dbo].[GetItemListSelection]
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[GetItemLotDetails]') and
OBJECTPROPERTY(id, N'IsProcedure') = 1)
drop procedure [dbo].[GetItemLotDetails]
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[GetItemTransCount]') and
OBJECTPROPERTY(id, N'IsProcedure') = 1)
drop procedure [dbo].[GetItemTransCount]
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[GetLotPickListForIssue]') and
OBJECTPROPERTY(id, N'IsProcedure') = 1)
drop procedure [dbo].[GetLotPickListForIssue]
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[GetMetaTableList]') and OBJECTPROPERTY(id,
N'IsProcedure') = 1)
drop procedure [dbo].[GetMetaTableList]
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[GetRecInvFacRpt]') and OBJECTPROPERTY(id,
N'IsProcedure') = 1)
drop procedure [dbo].[GetRecInvFacRpt]
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[GetRecInvManyInfo]') and
OBJECTPROPERTY(id, N'IsProcedure') = 1)
drop procedure [dbo].[GetRecInvManyInfo]
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[GetRecInvOneInfo]') and
OBJECTPROPERTY(id, N'IsProcedure') = 1)
drop procedure [dbo].[GetRecInvOneInfo]
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[GetRecInvRpt]') and OBJECTPROPERTY(id,
N'IsProcedure') = 1)
drop procedure [dbo].[GetRecInvRpt]
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[GetRecInvTree]') and OBJECTPROPERTY(id,
N'IsProcedure') = 1)
drop procedure [dbo].[GetRecInvTree]
GO
```

```
if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[GetStockSummaryAnyRpt]') and
OBJECTPROPERTY(id, N'IsProcedure') = 1)
drop procedure [dbo].[GetStockSummaryAnyRpt]
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[GetStockSummaryLotAnyRpt]') and
OBJECTPROPERTY(id, N'IsProcedure') = 1)
drop procedure [dbo].[GetStockSummaryLotAnyRpt]
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[GetStockSummaryLotRpt]') and
OBJECTPROPERTY(id, N'IsProcedure') = 1)
drop procedure [dbo].[GetStockSummaryLotRpt]
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[GetStockSummaryLotwise]') and
OBJECTPROPERTY(id, N'IsProcedure') = 1)
drop procedure [dbo].[GetStockSummaryLotwise]
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[GetStockSummaryRpt]') and
OBJECTPROPERTY(id, N'IsProcedure') = 1)
drop procedure [dbo].[GetStockSummaryRpt]
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[GetSupplierDetails]') and
OBJECTPROPERTY(id, N'IsProcedure') = 1)
drop procedure [dbo].[GetSupplierDetails]
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[GetSupplierList]') and OBJECTPROPERTY(id,
N'IsProcedure') = 1)
drop procedure [dbo].[GetSupplierList]
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[GetUnitOfMeasList]') and
OBJECTPROPERTY(id, N'IsProcedure') = 1)
drop procedure [dbo].[GetUnitOfMeasList]
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[GetWarehouseOneInfo]') and
OBJECTPROPERTY(id, N'IsProcedure') = 1)
drop procedure [dbo].[GetWarehouseOneInfo]
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[PostToStock_AdjInv]') and
OBJECTPROPERTY(id, N'IsProcedure') = 1)
drop procedure [dbo].[PostToStock_AdjInv]
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[PostToStock_IssueInv]') and
OBJECTPROPERTY(id, N'IsProcedure') = 1)
drop procedure [dbo].[PostToStock_IssueInv]
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[PostToStock_RecInv]') and
OBJECTPROPERTY(id, N'IsProcedure') = 1)
drop procedure [dbo].[PostToStock_RecInv]
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[TestDSQL]') and OBJECTPROPERTY(id,
N'IsProcedure') = 1)
drop procedure [dbo].[TestDSQL]
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[AdjInv]') and OBJECTPROPERTY(id,
N'IsUserTable') = 1)
drop table [dbo].[AdjInv]
GO
```

```
if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[AdjInvItems]') and OBJECTPROPERTY(id,
N'IsUserTable') = 1)
drop table [dbo].[AdjInvItems]
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[AdjType]') and OBJECTPROPERTY(id,
N'IsUserTable') = 1)
drop table [dbo].[AdjType]
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[Designation]') and OBJECTPROPERTY(id,
N'IsUserTable') = 1)
drop table [dbo].[Designation]
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[District]') and OBJECTPROPERTY(id,
N'IsUserTable') = 1)
drop table [dbo].[District]
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[Division]') and OBJECTPROPERTY(id,
N'IsUserTable') = 1)
drop table [dbo].[Division]
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[Employee]') and OBJECTPROPERTY(id,
N'IsUserTable') = 1)
drop table [dbo].[Employee]
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[FGroup]') and OBJECTPROPERTY(id,
N'IsUserTable') = 1)
drop table [dbo].[FGroup]
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[Facility]') and OBJECTPROPERTY(id,
N'IsUserTable') = 1)
drop table [dbo].[Facility]
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[FormAccess]') and OBJECTPROPERTY(id,
N'IsUserTable') = 1)
drop table [dbo].[FormAccess]
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[FormList]') and OBJECTPROPERTY(id,
N'IsUserTable') = 1)
drop table [dbo].[FormList]
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[GatePass]') and OBJECTPROPERTY(id,
N'IsUserTable') = 1)
drop table [dbo].[GatePass]
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[GatePassItems]') and OBJECTPROPERTY(id,
N'IsUserTable') = 1)
drop table [dbo].[GatePassItems]
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[Indent]') and OBJECTPROPERTY(id,
N'IsUserTable') = 1)
drop table [dbo].[Indent]
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[IndentDesig]') and OBJECTPROPERTY(id,
N'IsUserTable') = 1)
drop table [dbo].[IndentDesig]
```

```
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[IndentItems]') and OBJECTPROPERTY(id,
N'IsUserTable') = 1)
drop table [dbo].[IndentItems]
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[ItemGroup]') and OBJECTPROPERTY(id,
N'IsUserTable') = 1)
drop table [dbo].[ItemGroup]
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[ItemList]') and OBJECTPROPERTY(id,
N'IsUserTable') = 1)
drop table [dbo].[ItemList]
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[ItemListLot]') and OBJECTPROPERTY(id,
N'IsUserTable') = 1)
drop table [dbo].[ItemListLot]
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[MetaTable]') and OBJECTPROPERTY(id,
N'IsUserTable') = 1)
drop table [dbo].[MetaTable]
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[RecInv]') and OBJECTPROPERTY(id,
N'IsUserTable') = 1)
drop table [dbo].[RecInv]
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[RecInvItems]') and OBJECTPROPERTY(id,
N'IsUserTable') = 1)
drop table [dbo].[RecInvItems]
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[SQLLog]') and OBJECTPROPERTY(id,
N'IsUserTable') = 1)
drop table [dbo].[SQLLog]
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[SQLUpdate]') and OBJECTPROPERTY(id,
N'IsUserTable') = 1)
drop table [dbo].[SQLUpdate]
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[Supplier]') and OBJECTPROPERTY(id,
N'IsUserTable') = 1)
drop table [dbo].[Supplier]
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[UnitOfMeas]') and OBJECTPROPERTY(id,
N'IsUserTable') = 1)
drop table [dbo].[UnitOfMeas]
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[Warehouse]') and OBJECTPROPERTY(id,
N'IsUserTable') = 1)
drop table [dbo].[Warehouse]
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[tmpStockDetails]') and OBJECTPROPERTY(id,
N'IsUserTable') = 1)
drop table [dbo].[tmpStockDetails]
GO

CREATE TABLE [dbo].[AdjInv] (
        [InvNo] [int] NOT NULL ,
```

```
        [FacilityCode] [varchar] (4) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL ,
        [FromFacility] [varchar] (4) COLLATE SQL_Latin1_General_CP1_CI_AS NULL ,
        [AdjType] [smallint] NOT NULL ,
        [AdjRemarks] [varchar] (200) COLLATE SQL_Latin1_General_CP1_CI_AS NULL ,
        [PrepBy] [int] NOT NULL ,
        [PrepDate] [datetime] NOT NULL ,
        [AppBy1] [int] NOT NULL ,
        [AppDate1] [datetime] NOT NULL ,
        [IssuedBy] [int] NULL ,
        [IssuedDate] [datetime] NULL ,
        [bUpdated] [bit] NOT NULL
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[AdjInvItems] (
        [ItemSl] [int] IDENTITY (1, 1) NOT NULL ,
        [InvNo] [int] NOT NULL ,
        [ItemCode] [varchar] (10) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL ,
        [CurStockQty] [decimal](18, 0) NULL ,
        [IssueQty] [decimal](18, 0) NULL ,
        [MfgDate] [datetime] NULL ,
        [ExpDate] [datetime] NULL ,
        [LotNo] [varchar] (100) COLLATE SQL_Latin1_General_CP1_CI_AS NULL
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[AdjType] (
        [AdjType] [smallint] NOT NULL ,
        [AdjTypeName] [varchar] (40) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL ,
        [UsableIn] [smallint] NOT NULL ,
        [UsableOut] [smallint] NOT NULL ,
        [UnusableIn] [smallint] NOT NULL ,
        [UnusableOut] [smallint] NOT NULL
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[Designation] (
        [DesigCode] [smallint] NOT NULL ,
        [Designation] [varchar] (50) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[District] (
        [DistCode] [int] NOT NULL ,
        [DivCode] [smallint] NOT NULL ,
        [DistName] [varchar] (50) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[Division] (
        [DivCode] [smallint] NOT NULL ,
        [DivName] [varchar] (20) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[Employee] (
        [EmpCode] [int] NOT NULL ,
        [EmpName] [varchar] (100) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL ,
        [DesigCode] [smallint] NOT NULL ,
        [UserId] [varchar] (8) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL ,
        [UserPwd] [varchar] (8) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL ,
        [bActive] [bit] NOT NULL
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[FGroup] (
        [FGroupCode] [char] (1) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL ,
        [FGroupName] [varchar] (30) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL
) ON [PRIMARY]
```

```
GO

CREATE TABLE [dbo].[Facility] (
	[FacilityCode] [varchar] (4) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL ,
	[DistCode] [int] NOT NULL ,
	[FacilityName] [varchar] (100) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL ,
	[bShow] [bit] NOT NULL
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[FormAccess] (
	[FormId] [varchar] (20) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL ,
	[EmpCode] [int] NOT NULL ,
	[bAccess] [smallint] NOT NULL
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[FormList] (
	[FormId] [varchar] (20) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL ,
	[FormTitle] [varchar] (50) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[GatePass] (
	[GPassNo] [int] NOT NULL ,
	[DriverName] [varchar] (100) COLLATE SQL_Latin1_General_CP1_CI_AS NULL ,
	[VehicleNo] [varchar] (100) COLLATE SQL_Latin1_General_CP1_CI_AS NULL ,
	[GPassDate] [datetime] NOT NULL ,
	[AppBy1] [int] NULL ,
	[AppBy2] [int] NULL
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[GatePassItems] (
	[ItemSl] [int] IDENTITY (1, 1) NOT NULL ,
	[GPassNo] [int] NOT NULL ,
	[InvNo] [int] NOT NULL ,
	[InvType] [smallint] NOT NULL
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[Indent] (
	[InvNo] [int] NOT NULL ,
	[InvType] [smallint] NOT NULL ,
	[FacilityCode] [varchar] (4) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL ,
	[IndentNo] [varchar] (50) COLLATE SQL_Latin1_General_CP1_CI_AS NULL ,
	[PrepBy] [int] NOT NULL ,
	[PrepDate] [datetime] NOT NULL ,
	[AppBy1] [int] NOT NULL ,
	[AppDate1] [datetime] NOT NULL ,
	[IssuedBy] [int] NOT NULL ,
	[IssuedDate] [datetime] NOT NULL ,
	[IndentPrepBy] [varchar] (100) COLLATE SQL_Latin1_General_CP1_CI_AS NULL ,
	[IndentPrepDate] [datetime] NULL ,
	[IndentPrepDesig] [smallint] NULL ,
	[IndentAppBy1] [varchar] (100) COLLATE SQL_Latin1_General_CP1_CI_AS NULL ,
	[IndentAppDate1] [datetime] NULL ,
	[IndentAppDesig1] [smallint] NULL ,
	[bUpdated] [bit] NOT NULL ,
	[bDelivered] [bit] NOT NULL ,
	[Remarks] [varchar] (200) COLLATE SQL_Latin1_General_CP1_CI_AS NULL
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[IndentDesig] (
	[DesigCode] [smallint] NOT NULL ,
	[Designation] [varchar] (50) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL
) ON [PRIMARY]
GO
```

```
CREATE TABLE [dbo].[IndentItems] (
	[ItemSl] [int] IDENTITY (1, 1) NOT NULL ,
	[InvNo] [int] NOT NULL ,
	[InvType] [smallint] NOT NULL ,
	[ItemCode] [varchar] (10) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL ,
	[CurStockQty] [decimal](18, 0) NULL ,
	[ReqQty] [decimal](18, 0) NULL ,
	[IssueQty] [decimal](18, 0) NULL ,
	[MfgDate] [datetime] NULL ,
	[ExpDate] [datetime] NULL ,
	[LotNo] [varchar] (100) COLLATE SQL_Latin1_General_CP1_CI_AS NULL
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[ItemGroup] (
	[GroupCode] [char] (3) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL ,
	[GroupName] [varchar] (100) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[ItemList] (
	[ItemCode] [varchar] (10) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL ,
	[GroupCode] [char] (3) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL ,
	[ItemName] [varchar] (150) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL ,
	[InitStock] [decimal](18, 0) NOT NULL ,
	[InitStockUn] [decimal](18, 0) NOT NULL ,
	[UnitId] [smallint] NOT NULL ,
	[UsableQty] [decimal](18, 0) NOT NULL ,
	[UnusableQty] [decimal](18, 0) NOT NULL
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[ItemListLot] (
	[ItemListLotId] [int] IDENTITY (1, 1) NOT NULL ,
	[ItemCode] [varchar] (10) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL ,
	[LotNo] [varchar] (100) COLLATE SQL_Latin1_General_CP1_CI_AS NULL ,
	[MfgDate] [datetime] NULL ,
	[ExpDate] [datetime] NULL ,
	[InitLotQty] [decimal](18, 0) NULL ,
	[InitLotQtyUn] [decimal](18, 0) NULL ,
	[LotQty] [decimal](18, 0) NOT NULL ,
	[LotQtyUn] [decimal](18, 0) NULL
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[MetaTable] (
	[MetaTable] [varchar] (100) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL ,
	[MetaDesc] [varchar] (100) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[RecInv] (
	[InvNo] [int] NOT NULL ,
	[InvType] [smallint] NOT NULL ,
	[FacilityCode] [varchar] (4) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL ,
	[SupCode] [int] NULL ,
	[SupRef] [varchar] (30) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL ,
	[SupDate] [datetime] NOT NULL ,
	[FromFacility] [varchar] (4) COLLATE SQL_Latin1_General_CP1_CI_AS NULL ,
	[ReceiptDate] [datetime] NOT NULL ,
	[ReceivedBy] [int] NOT NULL ,
	[bUpdated] [bit] NOT NULL ,
	[BLNo] [varchar] (200) COLLATE SQL_Latin1_General_CP1_CI_AS NULL ,
	[BLDate] [datetime] NULL
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[RecInvItems] (
```

```
	[ItemSl] [int] IDENTITY (1, 1) NOT NULL ,
	[InvNo] [int] NOT NULL ,
	[InvType] [smallint] NOT NULL ,
	[ItemCode] [varchar] (10) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL ,
	[Qty] [decimal](18, 0) NULL ,
	[UnitPrice] [decimal](18, 0) NULL ,
	[MfgDate] [datetime] NULL ,
	[ExpDate] [datetime] NULL ,
	[LotNo] [varchar] (100) COLLATE SQL_Latin1_General_CP1_CI_AS NULL
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[SQLLog] (
	[LogSeq] [int] IDENTITY (1, 1) NOT NULL ,
	[UserName] [varchar] (30) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL ,
	[Terminal] [varchar] (30) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL ,
	[TStamp] [datetime] NOT NULL ,
	[SqlText] [varchar] (1000) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL ,
	[Copied] [bit] NOT NULL
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[SQLUpdate] (
	[LogSeq] [int] IDENTITY (1, 1) NOT NULL ,
	[UserName] [varchar] (8) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL ,
	[Terminal] [varchar] (30) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL ,
	[TStamp] [datetime] NOT NULL ,
	[SqlText] [varchar] (1000) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL ,
	[Updated] [bit] NOT NULL
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[Supplier] (
	[SupCode] [int] NOT NULL ,
	[SupName] [varchar] (50) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL ,
	[Address] [varchar] (150) COLLATE SQL_Latin1_General_CP1_CI_AS NULL ,
	[Phone] [varchar] (100) COLLATE SQL_Latin1_General_CP1_CI_AS NULL ,
	[Fax] [varchar] (100) COLLATE SQL_Latin1_General_CP1_CI_AS NULL ,
	[EMail] [varchar] (100) COLLATE SQL_Latin1_General_CP1_CI_AS NULL
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[UnitOfMeas] (
	[UnitId] [smallint] NOT NULL ,
	[UnitName] [varchar] (20) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[Warehouse] (
	[FacilityCode] [varchar] (4) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL ,
	[FacilityName] [varchar] (100) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL ,
	[Address] [varchar] (100) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL ,
	[InvPrepBy] [int] NULL ,
	[InvAppBy] [int] NULL ,
	[InvSupBy] [int] NULL
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[tmpStockDetails] (
	[UCode] [char] (32) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL ,
	[CurDate] [datetime] NOT NULL ,
	[GroupCode] [char] (3) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL ,
	[GroupName] [varchar] (100) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL ,
	[ItemCode] [varchar] (10) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL ,
	[ItemName] [varchar] (150) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL ,
	[Supplier] [varchar] (100) COLLATE SQL_Latin1_General_CP1_CI_AS NULL ,
	[Receiver] [varchar] (100) COLLATE SQL_Latin1_General_CP1_CI_AS NULL ,
	[InvoiceNo] [varchar] (100) COLLATE SQL_Latin1_General_CP1_CI_AS NULL ,
	[StockDate] [datetime] NULL ,
```

```
	[TotQtyIn] [decimal](18, 0) NULL ,
	[TotQtyOut] [decimal](18, 0) NULL ,
	[TotUnQtyIn] [decimal](18, 0) NULL ,
	[TotUnQtyOut] [decimal](18, 0) NULL ,
	[UnitName] [varchar] (20) COLLATE SQL_Latin1_General_CP1_CI_AS NULL ,
	[InvType] [varchar] (100) COLLATE SQL_Latin1_General_CP1_CI_AS NULL
) ON [PRIMARY]
GO

ALTER TABLE [dbo].[AdjInv] WITH NOCHECK ADD
	CONSTRAINT [PK_AdjInv] PRIMARY KEY  CLUSTERED
	(
		[InvNo]
	)  ON [PRIMARY]
GO

ALTER TABLE [dbo].[AdjInvItems] WITH NOCHECK ADD
	CONSTRAINT [PK_AdjInvItems] PRIMARY KEY  CLUSTERED
	(
		[ItemSl]
	)  ON [PRIMARY]
GO

ALTER TABLE [dbo].[AdjType] WITH NOCHECK ADD
	CONSTRAINT [PK_AdjType] PRIMARY KEY  CLUSTERED
	(
		[AdjType]
	)  ON [PRIMARY]
GO

ALTER TABLE [dbo].[Designation] WITH NOCHECK ADD
	CONSTRAINT [PK_Designation] PRIMARY KEY  CLUSTERED
	(
		[DesigCode]
	)  ON [PRIMARY]
GO

ALTER TABLE [dbo].[District] WITH NOCHECK ADD
	CONSTRAINT [PK_District] PRIMARY KEY  CLUSTERED
	(
		[DistCode]
	)  ON [PRIMARY]
GO

ALTER TABLE [dbo].[Division] WITH NOCHECK ADD
	CONSTRAINT [PK_Division] PRIMARY KEY  CLUSTERED
	(
		[DivCode]
	)  ON [PRIMARY]
GO

ALTER TABLE [dbo].[Employee] WITH NOCHECK ADD
	CONSTRAINT [PK_Employee] PRIMARY KEY  CLUSTERED
	(
		[EmpCode]
	)  ON [PRIMARY]
GO

ALTER TABLE [dbo].[FGroup] WITH NOCHECK ADD
	CONSTRAINT [PK_FGroup] PRIMARY KEY  CLUSTERED
	(
		[FGroupCode]
	)  ON [PRIMARY]
GO

ALTER TABLE [dbo].[Facility] WITH NOCHECK ADD
	CONSTRAINT [PK_Facility] PRIMARY KEY  CLUSTERED
	(
		[FacilityCode]
```

```
	) ON [PRIMARY]
GO

ALTER TABLE [dbo].[FormAccess] WITH NOCHECK ADD
	CONSTRAINT [PK_FormAccess] PRIMARY KEY  CLUSTERED
	(
		[FormId],
		[EmpCode]
	) ON [PRIMARY]
GO

ALTER TABLE [dbo].[FormList] WITH NOCHECK ADD
	CONSTRAINT [PK_FormList] PRIMARY KEY  CLUSTERED
	(
		[FormId]
	) ON [PRIMARY]
GO

ALTER TABLE [dbo].[GatePass] WITH NOCHECK ADD
	CONSTRAINT [PK_GatePass] PRIMARY KEY  CLUSTERED
	(
		[GPassNo]
	) ON [PRIMARY]
GO

ALTER TABLE [dbo].[GatePassItems] WITH NOCHECK ADD
	CONSTRAINT [PK_GatePassItems] PRIMARY KEY  CLUSTERED
	(
		[ItemSl]
	) ON [PRIMARY]
GO

ALTER TABLE [dbo].[Indent] WITH NOCHECK ADD
	CONSTRAINT [PK_Indent] PRIMARY KEY  CLUSTERED
	(
		[InvNo],
		[InvType]
	) ON [PRIMARY]
GO

ALTER TABLE [dbo].[IndentDesig] WITH NOCHECK ADD
	CONSTRAINT [PK_IndentDesig] PRIMARY KEY  CLUSTERED
	(
		[DesigCode]
	) ON [PRIMARY]
GO

ALTER TABLE [dbo].[IndentItems] WITH NOCHECK ADD
	CONSTRAINT [PK_IndentItems] PRIMARY KEY  CLUSTERED
	(
		[ItemSl]
	) ON [PRIMARY]
GO

ALTER TABLE [dbo].[ItemGroup] WITH NOCHECK ADD
	CONSTRAINT [PK_ItemGroup] PRIMARY KEY  CLUSTERED
	(
		[GroupCode]
	) ON [PRIMARY]
GO

ALTER TABLE [dbo].[ItemList] WITH NOCHECK ADD
	CONSTRAINT [PK_ItemList] PRIMARY KEY  CLUSTERED
	(
		[ItemCode]
	) ON [PRIMARY]
GO

ALTER TABLE [dbo].[ItemListLot] WITH NOCHECK ADD
```

```
        CONSTRAINT [PK_ItemListLot] PRIMARY KEY  CLUSTERED
        (
                [ItemListLotId]
        ) ON [PRIMARY]
GO

ALTER TABLE [dbo].[MetaTable] WITH NOCHECK ADD
        CONSTRAINT [PK_MetaTable] PRIMARY KEY  CLUSTERED
        (
                [MetaTable]
        ) ON [PRIMARY]
GO

ALTER TABLE [dbo].[RecInv] WITH NOCHECK ADD
        CONSTRAINT [PK_RecInv] PRIMARY KEY  CLUSTERED
        (
                [InvNo],
                [InvType]
        ) ON [PRIMARY]
GO

ALTER TABLE [dbo].[RecInvItems] WITH NOCHECK ADD
        CONSTRAINT [PK_RecInvItems] PRIMARY KEY  CLUSTERED
        (
                [ItemSl]
        ) ON [PRIMARY]
GO

ALTER TABLE [dbo].[SQLLog] WITH NOCHECK ADD
        CONSTRAINT [PK_SQLLog] PRIMARY KEY  CLUSTERED
        (
                [LogSeq]
        ) ON [PRIMARY]
GO

ALTER TABLE [dbo].[SQLUpdate] WITH NOCHECK ADD
        CONSTRAINT [PK_SQLUpdate] PRIMARY KEY  CLUSTERED
        (
                [LogSeq]
        ) ON [PRIMARY]
GO

ALTER TABLE [dbo].[Supplier] WITH NOCHECK ADD
        CONSTRAINT [PK_Supplier] PRIMARY KEY  CLUSTERED
        (
                [SupCode]
        ) ON [PRIMARY]
GO

ALTER TABLE [dbo].[UnitOfMeas] WITH NOCHECK ADD
        CONSTRAINT [PK_UnitOfMeas] PRIMARY KEY  CLUSTERED
        (
                [UnitId]
        ) ON [PRIMARY]
GO

ALTER TABLE [dbo].[Warehouse] WITH NOCHECK ADD
        CONSTRAINT [PK_Warehouse] PRIMARY KEY  CLUSTERED
        (
                [FacilityCode]
        ) ON [PRIMARY]
GO

ALTER TABLE [dbo].[AdjInv] WITH NOCHECK ADD
        CONSTRAINT [DF_AdjInv_bUpdated] DEFAULT (0) FOR [bUpdated]
GO

ALTER TABLE [dbo].[AdjType] WITH NOCHECK ADD
        CONSTRAINT [DF_AdjType_UsableIn] DEFAULT (0) FOR [UsableIn],
```

```
        CONSTRAINT [DF_AdjType_UsableOut] DEFAULT (0) FOR [UsableOut],
        CONSTRAINT [DF_AdjType_UnusableIn] DEFAULT (0) FOR [UnusableIn],
        CONSTRAINT [DF_AdjType_UnusableOut] DEFAULT (0) FOR [UnusableOut]
GO

ALTER TABLE [dbo].[Employee] WITH NOCHECK ADD
        CONSTRAINT [DF_Employee_EmpPwd] DEFAULT ('DFP') FOR [UserPwd],
        CONSTRAINT [DF_Employee_bActive] DEFAULT (1) FOR [bActive],
        CONSTRAINT [U_UserId] UNIQUE  NONCLUSTERED
        (
                [UserId]
        ) ON [PRIMARY]
GO

ALTER TABLE [dbo].[Facility] WITH NOCHECK ADD
        CONSTRAINT [DF_Facility_bShow] DEFAULT (1) FOR [bShow]
GO

ALTER TABLE [dbo].[FormAccess] WITH NOCHECK ADD
        CONSTRAINT [DF_FormAccess_bAccess] DEFAULT (0) FOR [bAccess]
GO

ALTER TABLE [dbo].[Indent] WITH NOCHECK ADD
        CONSTRAINT [DF_Indent_bUpdated] DEFAULT (0) FOR [bUpdated],
        CONSTRAINT [DF_Indent_bDelivered] DEFAULT (0) FOR [bDelivered]
GO

ALTER TABLE [dbo].[ItemList] WITH NOCHECK ADD
        CONSTRAINT [DF_ItemList_InitStock] DEFAULT (0) FOR [InitStock],
        CONSTRAINT [DF_ItemList_InitStockUn] DEFAULT (0) FOR [InitStockUn],
        CONSTRAINT [DF_ItemList_UnitId] DEFAULT (0) FOR [UnitId],
        CONSTRAINT [DF_ItemList_UsableQty] DEFAULT (0) FOR [UsableQty],
        CONSTRAINT [DF_ItemList_UnusableQty] DEFAULT (0) FOR [UnusableQty]
GO

ALTER TABLE [dbo].[RecInv] WITH NOCHECK ADD
        CONSTRAINT [DF_RecInv_bUpdated] DEFAULT (0) FOR [bUpdated]
GO

ALTER TABLE [dbo].[SQLLog] WITH NOCHECK ADD
        CONSTRAINT [DF_SQLLog_Copied] DEFAULT (0) FOR [Copied]
GO

ALTER TABLE [dbo].[SQLUpdate] WITH NOCHECK ADD
        CONSTRAINT [DF_SQLUpdate_Updated] DEFAULT (0) FOR [Updated]
GO

 CREATE  UNIQUE  INDEX [U_AdjTypeName] ON [dbo].[AdjType]([AdjTypeName]) ON [PRIMARY]
GO

 CREATE  UNIQUE  INDEX [U_Designation] ON [dbo].[Designation]([Designation]) ON [PRIMARY]
GO

 CREATE  UNIQUE  INDEX [U_DistName] ON [dbo].[District]([DistName]) ON [PRIMARY]
GO

 CREATE  UNIQUE  INDEX [U_DivName] ON [dbo].[Division]([DivName]) ON [PRIMARY]
GO

 CREATE  UNIQUE  INDEX [U_FacilityName] ON [dbo].[Facility]([FacilityName]) ON [PRIMARY]
GO

 CREATE  UNIQUE  INDEX [U_Designation] ON [dbo].[IndentDesig]([Designation]) ON [PRIMARY]
GO

 CREATE  UNIQUE  INDEX [U_GroupName] ON [dbo].[ItemGroup]([GroupName]) ON [PRIMARY]
GO

 CREATE  INDEX [IX_MetaTable] ON [dbo].[MetaTable]([MetaTable]) ON [PRIMARY]
```

```
GO

 CREATE  UNIQUE  INDEX [U_SupName] ON [dbo].[Supplier]([SupName]) ON [PRIMARY]
GO

 CREATE  UNIQUE  INDEX [U_UnitName] ON [dbo].[UnitOfMeas]([UnitName]) ON [PRIMARY]
GO

ALTER TABLE [dbo].[AdjInv] ADD
        CONSTRAINT [FK_AdjInv_AdjType] FOREIGN KEY
        (
                [AdjType]
        ) REFERENCES [dbo].[AdjType] (
                [AdjType]
        ),
        CONSTRAINT [FK_AdjInv_Facility] FOREIGN KEY
        (
                [FacilityCode]
        ) REFERENCES [dbo].[Facility] (
                [FacilityCode]
        ),
        CONSTRAINT [FK_AdjInv_Facility1] FOREIGN KEY
        (
                [FromFacility]
        ) REFERENCES [dbo].[Facility] (
                [FacilityCode]
        )
GO

ALTER TABLE [dbo].[AdjInvItems] ADD
        CONSTRAINT [FK_AdjInvItems_AdjInv] FOREIGN KEY
        (
                [InvNo]
        ) REFERENCES [dbo].[AdjInv] (
                [InvNo]
        )
GO

ALTER TABLE [dbo].[District] ADD
        CONSTRAINT [FK_District_Division] FOREIGN KEY
        (
                [DivCode]
        ) REFERENCES [dbo].[Division] (
                [DivCode]
        )
GO

ALTER TABLE [dbo].[Employee] ADD
        CONSTRAINT [FK_Employee_Designation] FOREIGN KEY
        (
                [DesigCode]
        ) REFERENCES [dbo].[Designation] (
                [DesigCode]
        )
GO

ALTER TABLE [dbo].[Facility] ADD
        CONSTRAINT [FK_Facility_District] FOREIGN KEY
        (
                [DistCode]
        ) REFERENCES [dbo].[District] (
                [DistCode]
        )
GO

ALTER TABLE [dbo].[FormAccess] ADD
        CONSTRAINT [FK_FormAccess_Employee] FOREIGN KEY
        (
                [EmpCode]
```

```
        ) REFERENCES [dbo].[Employee] (
                [EmpCode]
        ),
        CONSTRAINT [FK_FormAccess_FormList] FOREIGN KEY
        (
                [FormId]
        ) REFERENCES [dbo].[FormList] (
                [FormId]
        )
GO

ALTER TABLE [dbo].[GatePassItems] ADD
        CONSTRAINT [FK_GatePassItems_GatePass] FOREIGN KEY
        (
                [GPassNo]
        ) REFERENCES [dbo].[GatePass] (
                [GPassNo]
        ) ON UPDATE CASCADE ,
        CONSTRAINT [FK_GatePassItems_Indent] FOREIGN KEY
        (
                [InvNo],
                [InvType]
        ) REFERENCES [dbo].[Indent] (
                [InvNo],
                [InvType]
        )
GO

ALTER TABLE [dbo].[Indent] ADD
        CONSTRAINT [FK_Indent_Facility] FOREIGN KEY
        (
                [FacilityCode]
        ) REFERENCES [dbo].[Facility] (
                [FacilityCode]
        )
GO

ALTER TABLE [dbo].[IndentItems] ADD
        CONSTRAINT [FK_IndentItems_Indent] FOREIGN KEY
        (
                [InvNo],
                [InvType]
        ) REFERENCES [dbo].[Indent] (
                [InvNo],
                [InvType]
        )
GO

ALTER TABLE [dbo].[ItemList] ADD
        CONSTRAINT [FK_ItemList_ItemGroup] FOREIGN KEY
        (
                [GroupCode]
        ) REFERENCES [dbo].[ItemGroup] (
                [GroupCode]
        ),
        CONSTRAINT [FK_ItemList_UnitOfMeas] FOREIGN KEY
        (
                [UnitId]
        ) REFERENCES [dbo].[UnitOfMeas] (
                [UnitId]
        )
GO

ALTER TABLE [dbo].[ItemListLot] ADD
        CONSTRAINT [FK_ItemListLot_ItemList] FOREIGN KEY
        (
                [ItemCode]
        ) REFERENCES [dbo].[ItemList] (
                [ItemCode]
```

```
	) ON DELETE CASCADE  ON UPDATE CASCADE
GO

ALTER TABLE [dbo].[RecInv] ADD
	CONSTRAINT [FK_RecInv_Facility] FOREIGN KEY
	(
		[FacilityCode]
	) REFERENCES [dbo].[Facility] (
		[FacilityCode]
	),
	CONSTRAINT [FK_RecInv_Facility1] FOREIGN KEY
	(
		[FromFacility]
	) REFERENCES [dbo].[Facility] (
		[FacilityCode]
	),
	CONSTRAINT [FK_RecInv_Supplier] FOREIGN KEY
	(
		[SupCode]
	) REFERENCES [dbo].[Supplier] (
		[SupCode]
	)
GO

ALTER TABLE [dbo].[RecInvItems] ADD
	CONSTRAINT [FK_RecInvItems_RecInv] FOREIGN KEY
	(
		[InvNo],
		[InvType]
	) REFERENCES [dbo].[RecInv] (
		[InvNo],
		[InvType]
	)
GO


CREATE TRIGGER Insert_ItemListLot ON [dbo].[ItemList]
FOR INSERT
AS

declare @pItemCode as varchar(20)
declare @pInitQty as decimal
declare @pInitQtyUn as decimal

select @pItemCode=inserted.ItemCode, @pInitQty=inserted.InitStock, @pInitQtyUn=inserted.InitStockUn
from inserted

insert into ItemListLot (ItemCode, InitLotQty, InitLotQtyUn, LotQty, LotQtyUn) values
(@pItemCode, @pInitQty, @pInitQtyUn, isnull(@pInitQty,0), isnull(@pInitQtyUn,0))

GO
```

Stored Procedures

There are a total of 59 stored procedures in WIMS. Some stored procedures are used in data entry forms and some for report data source. The stored procedures used in data entry forms are described here. Stored procedures used in reports are discussed in the Reports section.

GetAdjInvManyInfo

Returns item details information from a specific adjustment invoice. In an adjustment invoice data entry form, when an adjustment invoice is selected from the left-side tree, this procedure is referred to as “get the item details information of the adjustment invoice.”
Parameters:
@InvNo—adjustment invoice number
Referenced In: frmAdjInv form

```
CREATE PROCEDURE dbo.GetAdjInvManyInfo
(
	@InvNo integer
)
AS

SELECT AdjInvItems.ItemSl, AdjInvItems.ItemCode, ItemList.ItemName, UnitOfMeas.UnitName,
AdjInvItems.IssueQty, AdjInvItems.ExpDate, AdjInvItems.LotNo, AdjInvItems.MfgDate,
AdjInvItems.CurStockQty
FROM UnitOfMeas INNER JOIN (ItemList INNER JOIN AdjInvItems ON ItemList.ItemCode =
AdjInvItems.ItemCode) ON UnitOfMeas.UnitId = ItemList.UnitId
where InvNo=@InvNo
order by ItemSl

GO
```

GetAdjInvOneInfo

Returns basic information from a specific adjustment invoice. In an adjustment invoice data entry form, when an adjustment invoice is selected from the left-side tree, this procedure is referred to as “get the basic information of the adjustment invoice.”
Parameters:
@InvNo—adjustment invoice number
Referenced In: frmAdjInv form

```
CREATE PROCEDURE dbo.GetAdjInvOneInfo
(
	@InvNo integer
)
AS

SELECT AdjInv.InvNo, AdjInv.FacilityCode, AdjInv.FromFacility,
AdjInv.PrepBy, AdjInv.PrepDate, AdjInv.AppBy1, AdjInv.AppDate1,
AdjInv.IssuedBy, AdjInv.IssuedDate,
AdjInv.AdjRemarks, AdjInv.AdjType, AdjInv.bUpdated
From AdjInv where InvNo=@InvNo

GO
```

GetAdjInvTree

This procedure returns a list of adjustment invoice numbers to appear in the tree of frmAdjInv form. A subset of data is returned based on start and end date criteria. Also, one can search a specific invoice can be done.
Parameters:
@SqlType—parameter to generate different types of output; 0: returns only invoice numbers, 1: returns adjustment type & invoice number, 2: returns date and invoice number, 999: find a specific invoice
@pSDate—start date from which invoices should be listed
@pEDate—end date from which invoices should be listed
@InvNo—adjustment invoice number
Referenced In: frmAdjInv form

```
CREATE PROCEDURE dbo.GetAdjInvTree
(
        @SqlType integer,
        @pSDate varchar(10),
        @pEDate varchar(10),
        @InvNo integer
)
AS

if @SqlType=0
        select InvNo from AdjInv
          where  PrepDate between @pSDate and @pEDate
          order by InvNo desc
else if @SqlType=1
        select AdjTypeName, InvNo
        from AdjInv inner join AdjType on AdjInv.AdjType=AdjType.AdjType
          where PrepDate between @pSDate and @pEDate
          order by AdjtypeName, InvNo desc
else if @SqlType=2
        select PrepDate, InvNo from AdjInv
          where PrepDate between @pSDate and @pEDate
          order by PrepDate desc, InvNo desc
else if @SqlType=9999
        select InvNo from AdjInv
          where InvNo=@InvNo
        and PrepDate between @pSDate and @pEDate

GO
```

GetAdjTypeList

Procedure returns list of adjustment types. This list of records appears in a combo to be selected during entry of adjustment invoice.
Parameters: None
Referenced In: frmAdjInv form

```
CREATE PROCEDURE dbo.GetAdjTypeList AS

select AdjType, AdjTypeName from AdjType order by AdjTypeName

GO
```

GetDesigList

Procedure returns list of employee designations. This list of records appears in a combo to be selected during entry of employee.
Parameters: None
Referenced In: frmEmployee form

```
CREATE PROCEDURE dbo.GetDesigList AS

select DesigCode, Designation from Designation order by Designation

GO
```

GetDistDetails

Procedure returns information for a single district. When a specific district is selected in district entry form, this procedure returns the data of a specific district to appear in the data entry fields.
Parameters:
@DistCode—specific district code
Referenced In: frmDistrict form

```
CREATE PROCEDURE dbo.GetDistDetails
(
        @DistCode integer
)
AS

select DistCode, DistName, DivCode
from District where DistCode=@DistCode

GO
```

GetDistTree

Procedure returns list of district codes to appear in the tree of frmDistrict form. Also, the list of records appears for selection of district during facility entry.
Parameters:
@SqlType—parameter to generate different types of output; 0: returns districts grouped by division, 1: returns districts ordering by name only
Referenced In: frmFacility form, frmDistrict form

```
CREATE PROCEDURE dbo.GetDistTree
(       @SqlType integer        ) AS

if @SqlType=0
   select District.DivCode, DivName, DistCode, DistName
   from District INNER JOIN Division ON District.DivCode=Division.DivCode
   order by DivName, DistName
else if @SqlType=1
    select DistCode, DistName
    from District
    order by DistName

GO
```

GetDivisionList

Procedure returns list of divisions of Bangladesh. This list of records is appears in a combo to be selected during entry of district.
Parameters: None
Referenced In: frmDistrict form

```
CREATE PROCEDURE dbo.GetDivisionList AS

select DivCode, DivName from Division order by DivName

GO
```

GetEmpAccess

With an employee code, procedure returns list of forms the user can access and his or her permissions. This information is retrieved as soon as an employee logs into the system, so that his or her access can be monitored.
Parameters:
@EmpCode—employee code
Referenced In: frmLogin form

```
CREATE PROCEDURE dbo.GetEmpAccess
(
        @EmpCode int
)
AS

select FormId, bAccess
from FormAccess where EmpCode=@EmpCode

GO
```

GetEmpList

Procedure returns list of employees. This list of records appears in a combo to be selected during entry of different types of receive and issue invoices.
Parameters: None
Referenced In: frmRecInv form, frmIssueIndent form, frmAdjInv form, frmGatePass form, frmIssueInv form, frmRecInvFac form

```
CREATE PROCEDURE dbo.GetEmpList AS

select EmpCode, EmpName from Employee order by EmpName

GO
```

GetEmpListForOpt

Procedure returns list of employees. This list of records appears in a combo to be selected during entry of different options.
Parameters: None
Referenced In: frmOptions form

```
CREATE PROCEDURE dbo.GetEmpListForOpt AS

select EmpCode, EmpName, Designation
from Employee inner join Designation on Employee.DesigCode=Designation.DesigCode
order by EmpName

GO
```

GetEmpListLogin

Procedure returns list of employees. This list of records appears in a combo during initial log on.
Parameters: None
Referenced In: frmLogin form

```
CREATE PROCEDURE dbo.GetEmpListLogin AS

select UserID, UserPwd, EmpCode, EmpName
from Employee order by UserID

GO
```

GetEmpManyInfo

With an employee code, procedure returns list of forms or reports with permissions of specific employees.
Parameters:
@EmpCode—employee code
@SqlType—returns list of forms or reports with employee permission
'FORMS'—returns form list
'REPORTS'—returns list of reports
Referenced In: frmEmployee form

```
CREATE PROCEDURE dbo.GetEmpManyInfo
(
        @EmpCode integer,
        @SqlType varchar(10)
)
AS

if @SqlType='FORMS'
        SELECT FormList.FormTitle, FormList.FormId, a.bAccess
        FROM FormList LEFT JOIN
          (select FormId, bAccess from FormAccess where EmpCode=@EmpCode) a
          ON FormList.FormId = a.FormId
          ORDER BY FormList.FormTitle
else if @SqlType='REPORTS'
        SELECT ReportList.ReportTitle, ReportList.ReportId, a.bAccess
          FROM ReportList LEFT JOIN
          (select ReportId, bAccess from ReportAccess where EmpCode=@EmpCode) a
          ON ReportList.ReportId = a.ReportId
          ORDER BY ReportList.ReportTitle
```

```
GO
```

GetEmpOneInfo

With an employee code, procedure returns basic employee information. When an employee is selected in a tree of frmEmployee form, this procedure provides the basic employee data to appear in the data entry fields.
Parameters:
@EmpCode—employee code
Referenced In: frmEmployee form

```
CREATE PROCEDURE dbo.GetEmpOneInfo
(
        @EmpCode integer
)
AS

select EmpCode, EmpName, DesigCode, UserId, UserPwd, bActive
from Employee where EmpCode=@EmpCode

GO
```

GetEmpTree

Procedure returns list of employee names to appear in the tree of frmEmployee form. Can also be used to find a specific employee.
Parameters:
@SqlType—parameter to generate different types of output. 0: returns employee code and name and sory by code; 1: returns employee code and name and sort by name, 999: given an employee name, finds an employee, 9999: given an employee code, finds an employee
Referenced In: frmEmployee form

```
CREATE PROCEDURE dbo.GetEmpTree
(
        @SqlType integer,
        @pEmpName varchar(100),
        @pEmpCode integer
)
AS

if @SqlType=0
        select EmpCode, EmpName from Employee
        order by EmpCode
else if @SqlType=1
        select EmpCode, EmpName from Employee
        order by EmpName
else if @SqlType=999
        select EmpCode from Employee
          where EmpName like @pEmpName
        order by EmpName
else if @SqlType=9999
        select EmpCode from Employee
          where EmpCode=@pEmpCode
GO
```

GetFGroupList

Procedure returns list of facility groups. This list of records appears in a combo to be selected when entering a facility.
Parameters: None
Referenced In: frmFacility form

```
CREATE PROCEDURE dbo.GetFGroupList AS

select FGroupCode, FGroupName from FGroup order by FGroupName

GO
```

GetFacilityDetails

Procedure returns information about a single facility. When a specific facility is selected on a facility entry form, this procedure returns the data of a specific facility to appear in the data entry fields.
Parameters:
@FacilityCode—specific facility code
Referenced In: frmFacility form

```
CREATE PROCEDURE dbo.GetFacilityDetails
(
        @FacilityCode varchar(4)
)
AS

select left(FacilityCode,1) as FGroup, right(FacilityCode,len(FacilityCode)-1) as FCode,
FacilityCode, DistCode, FacilityName, bShow
from Facility where FacilityCode=@FacilityCode

GO
```

GetFacilityList

Procedure returns list of facilities. This list of records appears in a combo to be selected during date entry or selection.
Parameters: None
Referenced In: frmIssueInv form, frmRecInvFac form, frmReports form, frmIssueIndent form, frmAdjInv form

```
CREATE PROCEDURE dbo.GetFacilityList AS

select FacilityCode, FacilityName from Facility order by FacilityName

GO
```

GetFacilityListForOpt

Procedure returns list of warehouses. This list of records appears in a combo to be selected during data entry or selection.
Parameters: None
Referenced In: frmOptions form

```
CREATE PROCEDURE dbo.GetFacilityListForOpt AS

select FacilityCode, FacilityName
from Facility where left(FacilityCode,1) in ('C','D','R')
order by FacilityName

GO
```

GetFacilityTree

This procedure returns list of facilities to appear in the tree of frmFacility form.
Parameters:
@SqlType—parameter to generate different types of output. 0: returns facility names grouped by district, 1: returns facility names grouped by facility group
Referenced In: frmFacility form

```
CREATE PROCEDURE dbo.GetFacilityTree
(
        @SqlType integer
)
AS

if @SqlType=0
   select Facility.DistCode, DistName, FacilityCode, FacilityName
   from Facility INNER JOIN District ON Facility.DistCode=District.DistCode
   order by DistName, FacilityName
else if @SqlType=1
    select FGroupCode, FGroupName, FacilityCode, FacilityName
    from FGroup INNER JOIN (
    select left(FacilityCode,1) as FGroup, FacilityCode, FacilityName
    from Facility
    ) FFacility on FGroup.FGroupCode=FFacility.FGroup
    order by FGroupName, FacilityName

GO
```

GetGPassManyInfo

Given a Gate Pass number, procedure returns information regarding invoices that are linked to this Gate Pass.
Parameters:
@GPassNo—Gate Pass number
Referenced In: frmGatePass form

```
CREATE PROCEDURE dbo.GetGPassManyInfo
(
        @GPassNo integer
)
AS
```

```
select ItemSl, InvNo, InvType from GatePassItems
where GPassNo=@GPassNo order by ItemSl

GO
```

GetGPassOneInfo

Given a Gate Pass number, procedure returns information regarding a Gate Pass.
Parameters:
@GPassNo—Gate Pass number
Referenced In: frmGatePass form

```
CREATE PROCEDURE dbo.GetGPassOneInfo
(
        @GPassNo integer
)
AS

select GPassNo, DriverName, VehicleNo, GPassDate, AppBy1, AppBy2
from GatePass where GPassNo=@GPassNo

GO
```

GetGPassTree

This procedure returns list of Gate Pass numbers to appear in the tree of frmGatePass form.
Parameters:
@SqlType—parameter to generate different types of output; 0: returns Gate Pass numbers only, 1: returns Gate Pass numbers grouped by date, 9999—given a Gate Pass number, find data for that Gate Pass
Referenced In: frmGatePass form

```
CREATE PROCEDURE dbo.GetGPassTree
(
        @SqlType integer,
        @pSDate varchar(10),
        @pEDate varchar(10),
        @GPassNo integer
)
AS

if @SqlType=0
        select GPassNo from GatePass
          where GPassDate between @pSDate and @pEDate
          order by GPassNo desc
else if @SqlType=1
        select GPassDate, GPassNo from GatePass
          where GPassDate between @pSDate and @pEDate
          order by GPassDate desc, GPassNo desc
else if @SqlType=9999
        select GPassNo from GatePass
          where GPassDate between @pSDate and @pEDate
        and GPassNo=@GPassNo

GO
```

GetGroupDetails

Procedure returns information for a single item group. When a specific item group is selected in item group entry form, this procedure returns the data of a specific item group to appear in the data entry fields.
Parameters:
@GroupCode—specific group code
Referenced In: frmItemGroup form

```
CREATE PROCEDURE dbo.GetGroupDetails
(
        @GroupCode char(3)
)
AS

select GroupCode, GroupName from ItemGroup
where GroupCode=@GroupCode

GO
```

GetGroupList

Procedure returns list of item groups. This list of records appears in a combo to be selected during date entry or selection.
Parameters:
@SqlType—criteria to produce different list of records; 0: selects only group code and names, 1: together with group codes and names, adds an extra item ‘ALL‘ for selection during report criteria selection
Referenced In: frmItemGroup form, frmStockSummary form, frmReports form, frmItemList form, frmItemGroup form

```
CREATE PROCEDURE dbo.GetGroupList
(
        @SqlType integer
)
AS

if @SqlType=0
        select GroupCode, GroupName from ItemGroup
        order by GroupName
if @SqlType=1
        select distinct '(ALL)' as GroupCode, '(ALL)' as GroupName from  ItemGroup
          union all
        select GroupCode, GroupName from ItemGroup
        order by GroupName

GO
```

GetIndentDesigList

Procedure returns list of designation of indentors. This list of records appears in a combo to be selected during date entry.
Parameters: None
Referenced In: frmIssueInv form, frmIssueIndent form

```
CREATE PROCEDURE dbo.GetIndentDesigList AS

select DesigCode, Designation from IndentDesig order by Designation

GO
```

GetInvForGPass

Procedure returns list of invoices ready to be delivered. This list of records appears for pickup during Gate Pass entry.
Parameters: None
Referenced In: frmSelectInv form

```
CREATE PROCEDURE dbo.GetInvForGPass AS

SELECT Indent.FacilityCode, FacilityName, InvType, Indent.InvNo
FROM Indent inner join Facility on Indent.FacilityCode=Facility.FacilityCode
where bDelivered=0 and bUpdated=1
order by 1, 2, 3

GO
```

GetIssueInvManyInfo

Given an invoice number, procedure returns item details of the specific issue voucher.
Parameters:
@SqlType—different list of records depending on value; 0: returns item records for indent invoice, 1: returns item records for push issue invoice
@InvType—type of invoice; 1: push invoice, 2: indent invoice
@InvNo—issue invoice number
Referenced In: frmIssueInv form, frmIssueIndent form

```
CREATE PROCEDURE dbo.GetIssueInvManyInfo
(
        @SqlType integer,
        @InvType integer, @InvNo integer
)
AS

if @SqlType=0
        SELECT IndentItems.ItemSl, IndentItems.ItemCode, ItemList.ItemName,
UnitOfMeas.UnitName,
        IndentItems.ReqQty, IndentItems.IssueQty, IndentItems.ExpDate, IndentItems.LotNo,
IndentItems.MfgDate, IndentItems.CurStockQty
        FROM UnitOfMeas INNER JOIN (ItemList INNER JOIN IndentItems ON ItemList.ItemCode =
        IndentItems.ItemCode) ON UnitOfMeas.UnitId = ItemList.UnitId
        where InvType=@InvType and InvNo=@InvNo
        order by ItemSl
if @SqlType=1
        SELECT IndentItems.ItemSl, IndentItems.ItemCode, ItemList.ItemName,
UnitOfMeas.UnitName,
         IndentItems.ReqQty, IndentItems.IssueQty, IndentItems.ExpDate, IndentItems.LotNo,
IndentItems.MfgDate, IndentItems.CurStockQty
         FROM UnitOfMeas INNER JOIN (ItemList INNER JOIN IndentItems ON ItemList.ItemCode =
         IndentItems.ItemCode) ON UnitOfMeas.UnitId = ItemList.UnitId
        where InvType=@InvType and InvNo=@InvNo
        order by ItemSl

GO
```

GetIssueInvOneInfo

Given an invoice number, procedure returns data from the issue invoice master table.
Parameters:
@SqlType—different list of records depending on value; 0: returns data for indent invoice, 1: returns data for push issue invoice
@InvType—type of invoice; 1: push invoice, 2: indent invoice
@InvNo—issue invoice number
Referenced In: frmIssueInv form, frmIssueIndent form

```
CREATE PROCEDURE dbo.GetIssueInvOneInfo
(
        @SqlType integer,
        @InvType integer, @InvNo integer
)
AS

if @SqlType=0
        SELECT Indent.InvNo, Indent.FacilityCode, Indent.IndentNo, Indent.IndentPrepBy,
        Indent.IndentPrepDesig, Indent.IndentPrepDate, Indent.IndentAppBy1,
Indent.IndentAppDesig1,
        Indent.IndentAppDate1, Indent.PrepBy, Indent.PrepDate, Indent.AppBy1, Indent.AppDate1,
        Indent.IssuedBy, Indent.IssuedDate, Indent.bUpdated, Indent.Remarks
        From Indent where InvType=@InvType and InvNo=@InvNo
else if @SqlType=1
        SELECT Indent.InvNo, Indent.FacilityCode,
          Indent.PrepBy, Indent.PrepDate, Indent.AppBy1, Indent.AppDate1,
          Indent.IssuedBy, Indent.IssuedDate, Indent.bUpdated, Indent.Remarks
        From Indent where InvType=@InvType and InvNo=@InvNo

GO
```

GetIssueInvTree

This procedure returns list of issue invoice numbers to appear in the tree of issue invoice entry forms.
Parameters:
@SqlType—parameter to generate different types of output; 0: returns only invoice numbers, 1: returns invoice numbers grouped by facility, 2: returns invoice numbers grouped by date, 9999: given an invoice number, find data for that invoice
Referenced In: frmIssueInv form, frmIssueIndent form

```
CREATE PROCEDURE dbo.GetIssueInvTree
(
        @SqlType integer,
        @InvType integer,
        @pSDate varchar(10),
        @pEDate varchar(10),
        @InvNo integer
)
AS

if @SqlType=0
        select InvNo from Indent
          where InvType=@InvType
        and PrepDate between @pSDate and @pEDate
          order by InvNo desc
else if @SqlType=1
        select FacilityName, InvNo
        from Indent inner join Facility on Indent.FacilityCode=Facility.FacilityCode
```

```
          where InvType=@InvType
         and PrepDate between @pSDate and @pEDate
          order by FacilityName, InvNo desc
else if @SqlType=2
         select PrepDate, InvNo from Indent
          where InvType=@InvType
         and PrepDate between @pSDate and @pEDate
          order by PrepDate desc, InvNo desc
else if @SqlType=9999
         select InvNo from Indent
          where InvType=@InvType and InvNo=@InvNo
         and PrepDate between @pSDate and @pEDate
          order by InvNo desc

GO
```

GetItemCurStockQty

Given batch information of a specific product, this procedure returns the current available quantity of that item in stock.
Parameters:
@ItemCode—item code
@LotNo—lot/batch number
@MfgDate—manufacturing date of the batch
@ExpDate—expiry date of the batch
Referenced In: frmAdjInv form, frmIssueInv form, frmIssueIndent form

```
CREATE PROCEDURE dbo.GetItemCurStockQty
(
        @ItemCode varchar(100),
        @LotNo varchar(100),
        @MfgDate varchar(100),
        @ExpDate varchar(100)
)
AS

        select LotQty, LotQtyUn
        from ItemListLot
        where ItemCode=@ItemCode and LotNo=@LotNo and ExpDate=@ExpDate and
MfgDate=@MfgDate

GO
```

GetItemDetailsNew

This procedure returns a list of items used during data entry or item selection during report printing.
Parameters:
@GroupCode—item group code
@SqlType—returns different set of records; 0: returns a list of items for entry/editing, 1: returns a list of items for selection while report is printing
@OrderBy—records will be sorted by this field
@ItemName—can filter a subset of items based on this string pattern
Referenced In: frmItemList form, frmReports form

```
CREATE PROCEDURE dbo.GetItemDetailsNew
(
        @GroupCode char(10), @SqlType integer, @OrderBy integer, @ItemName varchar(150)
)
```

```
AS
declare @StrSql nvarchar(2000)
declare @StrCriteria nvarchar(1000)

if @SqlType=0
        begin
        set @StrSql = 'select ItemList.ItemCode, ItemList.ItemName, ItemList.InitStock,
ItemList.InitStockUn, ' +
                    'ItemList.UnitId, UnitOfMeas.UnitName ' +
                    'from ItemList inner join UnitOfMeas on ItemList.UnitId=UnitOfMeas.UnitId '

        set @StrCriteria = ''
        if @GroupCode <> '(ALL)'
                set @StrCriteria = ' GroupCode='  + char(39) + @GroupCode + char(39)
        if isnull(@ItemName,'') <> ''
        begin
                if @StrCriteria <> ''
                        set @StrCriteria = @StrCriteria + ' and '
                set @StrCriteria = @StrCriteria + ' ItemName like '  + char(39) + '%' + @ItemName
+ '%' + char(39)
        end
        if @StrCriteria <> ''
                set @StrSql = @StrSql + ' where ' + @StrCriteria
        if @OrderBy=0
                set @StrSql = @StrSql + ' order by ItemList.ItemCode'
        else if @OrderBy=1
                set @StrSql = @StrSql + ' order by ItemList.ItemName'

        exec sp_executesql @StrSql
        end
else if @SqlType=1
        select ItemName, ItemCode
        from ItemList
        where GroupCode=@GroupCode
        order by ItemName

GO
```

GetItemListSelection

This procedure returns a list of items used during invoice entry.
Parameters:
@SqlType—returns different set of records; 0: returns a list of items by item name, 1: returns a list of items with current usable stock quantity, 2: returns a list of items with non-zero stock, 3: select item list used during adjustment invoice entry, 4: returns item list with unusable quantity in stock
Referenced In: frmRecInv form, frmAdjInv form, frmRecInvFac form

```
CREATE PROCEDURE dbo.GetItemListSelection
(
        @SqlType integer
)
AS

if @SqlType=0
        select ItemName, ItemCode, UnitName
          from ItemList LEFT JOIN UnitOfMeas ON ItemList.UnitId=UnitOfMeas.UnitId
          order by ItemName
else if @SqlType=1
        select ItemName, ItemCode, UnitName, isnull(UsableQty,0) as NZUsableQty
```

```
          from ItemList LEFT JOIN UnitOfMeas ON ItemList.UnitId=UnitOfMeas.UnitId
          order by ItemName
else if @SqlType=2
        select ItemName, ItemCode, UnitName, isnull(UsableQty,0) as NZUsableQty
          from ItemList LEFT JOIN UnitOfMeas ON ItemList.UnitId=UnitOfMeas.UnitId
          where UsableQty > 0
          order by ItemName
else if @SqlType=3
        select ItemName, '' as ItemListLotId, ItemCode, '' as CLotNo, '' as CMfgDate, '' as CExpDate, ''
as LotQty, UnitName
          from ItemList LEFT JOIN UnitOfMeas ON ItemList.UnitId=UnitOfMeas.UnitId
          order by ItemName
else if @SqlType=4
        select ItemName, ItemCode, UnitName, isnull(UnUsableQty,0) as NZUnUsableQty
          from ItemList LEFT JOIN UnitOfMeas ON ItemList.UnitId=UnitOfMeas.UnitId
          where UnusableQty > 0
          order by ItemName

GO
```

GetItemLotDetails

Given a product code, this procedure returns the lotwise details of that particular item.
Parameters:
@ItemCode—item code
Referenced In: frmItemListLot form

```
CREATE PROCEDURE dbo.GetItemLotDetails
(       @ItemCode varchar(100)     )  AS

        select ItemListLotId, LotNo, MfgDate, ExpDate, LotQty, LotQtyUn, InitLotQty, InitLotQtyUn
        from ItemListLot
        where ItemCode=@ItemCode
        order by ExpDate desc

GO
```

GetItemTransCount

This procedure determines whether a specific item has transactions, also returns the transaction count. Used during item data entry, if TransCount > 0, cannot edit total initial quantity and lotwise initial quantity of item.
Parameters:
@ItemCode—item code
@TransCount—returns number of transactions of the item; 0 otherwise
Referenced In: frmItemListLot form, frmItemList form

```
CREATE PROCEDURE dbo.GetItemTransCount
(
        @ItemCode varchar(100), @TransCount integer output
)
AS

declare @tmpCount integer, @mainCount integer

        set @mainCount=0

        select @tmpCount=count(ItemSl) from RecInvItems where ItemCode=@ItemCode
```

```
	set @mainCount = @mainCount + @tmpCount

	select @tmpCount=count(ItemSl) from IndentItems where ItemCode=@ItemCode
	set @mainCount = @mainCount + @tmpCount

	select @tmpCount=count(ItemSl) from AdjInvItems where ItemCode=@ItemCode
	set @mainCount = @mainCount + @tmpCount

	set @TransCount = @mainCount

GO
```

GetLotPickListForIssue

This procedure determines whether a specific item has transactions, also returns the transaction count.
Parameters:
@SqlType—returns set of records, 0 -
@GroupCode—filters records by group
Referenced In: frmAdjInv form, frmIssueInv form, frmIssueIndent form

```
CREATE PROCEDURE dbo.GetLotPickListForIssue
(
	@SqlType integer,
	@GroupCode varchar(100)
)
AS

if @SqlType=0
	if @GroupCode='(ALL)'
		select ItemName, ItemListLotId, ItemListLot.ItemCode, isnull(LotNo,' ') as CLotNo,
		isnull(convert(char(10),MfgDate,103),'') as CMfgDate,
isnull(convert(char(10),ExpDate,103),'') as CExpDate, LotQty, UnitName
		from ItemListLot inner join (ItemList INNER JOIN UnitOfMeas on
ItemList.UnitId=UnitOfMeas.UnitId) on ItemListLot.ItemCode=ItemList.ItemCode
		where LotQty > 0
		order by ItemName, ExpDate
	else
		select ItemName, ItemListLotId, ItemListLot.ItemCode, LotNo, MfgDate, ExpDate,
LotQty
		from ItemListLot inner join ItemList on ItemListLot.ItemCode=ItemList.ItemCode
		where GroupCode=@GroupCode and LotQty > 0
		order by ItemName, ExpDate
else if @SqlType=1
	if @GroupCode='(ALL)'
		select ItemName, ItemListLotId, ItemListLot.ItemCode, isnull(LotNo,' ') as CLotNo,
		isnull(convert(char(10),MfgDate,103),'') as CMfgDate,
isnull(convert(char(10),ExpDate,103),'') as CExpDate, LotQtyUn, UnitName
		from ItemListLot inner join (ItemList INNER JOIN UnitOfMeas on
ItemList.UnitId=UnitOfMeas.UnitId) on ItemListLot.ItemCode=ItemList.ItemCode
		where LotQtyUn > 0
		order by ItemName, ExpDate
	else
		select ItemName, ItemListLotId, ItemListLot.ItemCode, LotNo, MfgDate, ExpDate,
LotQtyUn
		from ItemListLot inner join ItemList on ItemListLot.ItemCode=ItemList.ItemCode
		where GroupCode=@GroupCode and LotQtyUn > 0
		order by ItemName, ExpDate

GO
```

GetMetaTableList

Get the list of lookup tables to be edited using lookup tables list.
Parameters: None
Referenced In: frmMetaTables form

```
CREATE PROCEDURE dbo.GetMetaTableList AS

select MetaTable, MetaDesc from MetaTable order by MetaTable

GO
```

GetRecInvManyInfo

This procedure returns received item list information from a specific receive invoice.
Parameters:
@InvType—type of invoice; 1: external receive, 2: receive from another warehouse
@InvNo—specific invoice number
Referenced In: frmRecInv form, frmRecInvFac form

```
CREATE PROCEDURE dbo.GetRecInvManyInfo
(
        @InvType integer, @InvNo integer
)
AS

select RecInvItems.ItemSl, RecInvItems.ItemCode, ItemList.ItemName, UnitOfMeas.UnitName,
RecInvItems.Qty, RecInvItems.UnitPrice, RecInvItems.ExpDate, RecInvItems.LotNo,
RecInvItems.MfgDate
from UnitOfMeas INNER JOIN (ItemList INNER JOIN RecInvItems ON
ItemList.ItemCode = RecInvItems.ItemCode) ON UnitOfMeas.UnitId = ItemList.UnitId
where InvType=@InvType and InvNo=@InvNo
order by ItemSl

GO
```

GetRecInvOneInfo

This procedure returns basic information regarding a specific receive invoice.
Parameters:
@InvType—type of invoice; 1: external receive, 2: receive from another warehouse
@InvNo—specific invoice number
Referenced In: frmRecInv form, frmRecInvFac form

```
CREATE PROCEDURE dbo.GetRecInvOneInfo
(
        @InvType integer, @InvNo integer
)
AS

select InvNo, InvType, FacilityCode, SupCode, SupRef, SupDate,
FromFacility, ReceiptDate, ReceivedBy, bUpdated,  BLNo,  BLDate
from RecInv where InvType=@InvType and InvNo=@InvNo

GO
```

GetRecInvTree

This procedure returns a list of receive invoice numbers to appear in the tree of frmRecInv and frmRecInvFac form. A subset of data is returned based on start and end date criteria. Also, a specific invoice can be searched.
Parameters:
@SqlType—parameter to generate different types of output; 0: returns only invoice numbers, 1: returns date & invoice number, 9999: find a specific invoice
@pSDate—start date from which invoices should be listed
@pEDate—end date from which invoices should be listed
@InvNo—receive invoice number
Referenced In: frmRecInv form, frmrecInvFac form

```
CREATE PROCEDURE dbo.GetRecInvTree
(
        @SqlType integer,
        @InvType integer,
        @pSDate varchar(10),
        @pEDate varchar(10),
        @InvNo integer
)
AS

if @SqlType=0
        select InvNo from RecInv
          where InvType=@InvType
        and ReceiptDate between @pSDate and @pEDate
          order by InvNo desc
else if @SqlType=1
        select ReceiptDate, InvNo from RecInv
          where InvType=@InvType
        and ReceiptDate between @pSDate and @pEDate
          order by ReceiptDate desc, InvNo desc
else if @SqlType=9999
        select InvNo from RecInv
          where InvType=@InvType and InvNo=@InvNo
        and ReceiptDate between @pSDate and @pEDate
          order by InvNo desc

GO
```

GetStockSummaryLotwise

This procedure returns lotwise stock status of a specific item on a specific date.
Parameters:
@pItem—item code for which to generate lotwise information
@pNonZero—?
@pBalDate—stock status as of this date
Referenced In: frmStockLotSummary form

```
CREATE PROCEDURE dbo.GetStockSummaryLotwise
(
        @pItem varchar(10),
        @pNonZero integer,
        @pBalDate varchar(20)
)
```

```
 AS

declare @StrSql nvarchar(2000)
declare @Criteria nvarchar(500)

set @StrSql =
'select ItemList.GroupCode, ItemGroup.GroupName, ItemList.ItemCode, ItemList.ItemName,
LotNo, MfgDate, ExpDate,
sum(QtyIn-QtyOut) as UsableQty, sum(UnQtyIn-UnQtyOut) as UnusableQty, UnitName ' +
'from UnitOfMeas inner join (ItemGroup inner join (ItemList inner join (' +
'select ItemCode, LotNo, MfgDate, ExpDate,  InitLotQty as QtyIn, 0 as QtyOut, InitLotQtyUn as
UnQtyIn, 0 as UnQtyOut ' +
'from ItemListLot  ' +
'UNION ALL ' +
'select ItemCode, LotNo, MfgDate, ExpDate, sum(Qty) as QtyIn, 0 as QtyOut, 0 as UnQtyIn, 0 as
UnQtyOut ' +
'from RecInv INNER JOIN RecInvItems ON (RecInv.InvNo = RecInvItems.InvNo) AND (RecInv.InvType
= RecInvItems.InvType)  ' +
'where bUpdated=1 and ReceiptDate <= ' + char(39) +  @pBalDate + char(39) + ' group by
ItemCode, LotNo, MfgDate, ExpDate  UNION ALL ' +
'select ItemCode, LotNo, MfgDate, ExpDate,  0 as QtyIn, sum(IssueQty) as QtyOut, 0 as UnQtyIn, 0
as UnQtyOut ' +
'from Indent INNER JOIN IndentItems ON (Indent.InvNo = IndentItems.InvNo) AND (Indent.InvType
= IndentItems.InvType)  ' +
'where  bUpdated=1 and IssuedDate <= ' + char(39) + @pBalDate + char(39) + ' and IssueQty is not
NULL group by ItemCode, LotNo, MfgDate, ExpDate  UNION ALL ' +
'select ItemCode, LotNo, MfgDate, ExpDate,  sum(IssueQty*usablein) as QtyIn,
sum(IssueQty*usableout) as QtyOut, sum(IssueQty*unusablein) as UnQtyIn,
sum(IssueQty*unusableout) as UnQtyOut ' +
'from AdjType INNER JOIN (AdjInv INNER JOIN AdjInvItems ON (AdjInv.InvNo = AdjInvItems.InvNo))
ON AdjType.AdjType=AdjInv.AdjType  ' +
'where  bUpdated=1 and IssuedDate <= ' + char(39) + @pBalDate + char(39) + ' group by
ItemCode, LotNo, MfgDate, ExpDate ' +
') a  on a.ItemCode = ItemList.ItemCode) on ItemGroup.GroupCode = ItemList.GroupCode) on
UnitOfMeas.UnitId = ItemList.UnitId '

if isnull(@pItem,'') <> ''
        set @StrSql = @StrSql +  ' where ItemList.ItemCode=' + char(39)  + @pItem + char(39)
set @StrSql = @StrSql + ' group by ItemList.GroupCode, ItemGroup.GroupName, ItemList.ItemName,
UnitName, ItemList.ItemCode,LotNo, MfgDate, ExpDate '

exec sp_executesql @StrSql

GO
```

GetSupplierDetails

When provided the supplier code, this procedure returns detailed information about a supplier.
Parameters:
@SupCode—supplier code
Referenced In: frmSupplier form

```
CREATE PROCEDURE dbo.GetSupplierDetails
(
        @SupCode int
)
AS

select SupCode, SupName, Address, Phone, Fax, EMail from supplier
where SupCode=@SupCode
```

GO

GetSupplierList

This procedure returns the list of suppliers so it can appear in a selection list during data entry.
Parameters: None
Referenced In: frmSupplier form, frmRecInv form

```
CREATE PROCEDURE dbo.GetSupplierList AS

select SupCode, SupName from Supplier order by SupName

GO
```

GetUnitOfMeasList

This procedure returns the list of units of measurement so it can appear in a selection list during data entry.
Parameters: None
Referenced In: frmItemList form

```
CREATE PROCEDURE dbo.GetUnitOfMeasList AS

select UnitId, UnitName from UnitOfMeas
order by UnitName

GO
```

GetWarehouseOneInfo

This procedure returns the data regarding current warehouse.
Parameters: None
Referenced In: frmOptions form

```
CREATE PROCEDURE dbo.GetWarehouseOneInfo AS

select FacilityCode, FacilityName, Address, InvPrepBy, InvAppBy, InvSupBy
from Warehouse

GO
```

PostToStock_AdjInv

This procedure deducts the item quantities on the specific adjustment invoice from stock. Total number of records updated is returned. Total stock quantity is updated in ItemList table, and lotwise item quantity is updated in ItemListLot table.
Parameters:
@InvNo—adjustment invoice number
@Adjtype—type of adjustment
@RecCount—total number of records updated
Referenced In: frmAdjInv form

```
CREATE PROCEDURE dbo.PostToStock_AdjInv
(
```

```
        @InvNo integer,
        @AdjType integer,
        @RecCount integer output
)
AS

begin transaction

declare @ItemCode varchar(10), @IssueQty decimal, @LotNo varchar(100), @MfgDate datetime,
@ExpDate datetime
declare @tmpCount integer, @UpdCount integer
declare @ItemUsableQty integer, @ItemUnusableQty integer

declare c_AdjInvItems cursor for
select ItemCode, IssueQty, LotNo, MfgDate, ExpDate from AdjInvItems
where InvNo=@InvNo

open c_AdjInvItems

set @UpdCount=0
fetch next from c_AdjInvItems into @ItemCode,@IssueQty,@LotNo,@Mfgdate,@ExpDate
while @@FETCH_STATUS=0
begin
        set @tmpCount=0

        select @ItemUsableQty = UsableQty, @ItemUnusableQty = UnusableQty from ItemList where
ItemCode=@ItemCode
        if @AdjType=2 or @AdjType=5
           begin
                if isnull(@ItemUsableQty,0)-isnull(@IssueQty,0) < 0
                   begin
                        rollback
                        set @RecCount = -1
                        return
                   end
           end
        if @AdjType=6
           begin
                if isnull(@ItemUnusableQty,0)-isnull(@IssueQty,0) < 0
                   begin
                        rollback
                        set @RecCount = -1
                        return
                   end
           end

        update ItemList set UsableQty =
                case
                  when @AdjType=1 or @AdjType=3 then isnull(UsableQty,0)+isnull(@IssueQty,0)
                  when @AdjType=2 or @AdjType=5 then isnull(UsableQty,0)-isnull(@IssueQty,0)
                  else isnull(UsableQty,0)
                end
        ,UnusableQty =
                case
                  when @AdjType=4 or @AdjType=5 then isnull(UnusableQty,0)+isnull(@IssueQty,0)
                  when @AdjType=6 then isnull(UnusableQty,0)-isnull(@IssueQty,0)
                  else isnull(UnusableQty,0)
                end
         where ItemCode=@ItemCode
        set @tmpCount = @@ROWCOUNT
```

```
        if @LotNo is null and @ExpDate is null and @MfgDate is null
                update ItemListLot set LotQty =
                        case
                          when @AdjType=1 or @AdjType=3 then
isnull(LotQty,0)+isnull(@IssueQty,0)
                          when @AdjType=2 or @AdjType=5 then isnull(LotQty,0)-isnull(@IssueQty,0)
                          else isnull(LotQty,0)
                        end
                        ,LotQtyUn =
                        case
                          when @AdjType=4 or @AdjType=5 then
isnull(LotQtyUn,0)+isnull(@IssueQty,0)
                          when @AdjType=6 then isnull(LotQtyUn,0)-isnull(@IssueQty,0)
                          else isnull(LotQtyUn,0)
                        end
                where ItemCode=@ItemCode and LotNo is null and ExpDate is null and MfgDate is null
        else if  @LotNo is not null and @ExpDate is null and @MfgDate is null
                update ItemListLot set LotQty =
                        case
                          when @AdjType=1 or @AdjType=3 then
isnull(LotQty,0)+isnull(@IssueQty,0)
                          when @AdjType=2 or @AdjType=5 then isnull(LotQty,0)-isnull(@IssueQty,0)
                          else isnull(LotQty,0)
                        end
                        ,LotQtyUn =
                        case
                          when @AdjType=4 or @AdjType=5 then
isnull(LotQtyUn,0)+isnull(@IssueQty,0)
                          when @AdjType=6 then isnull(LotQtyUn,0)-isnull(@IssueQty,0)
                          else isnull(LotQtyUn,0)
                        end
                where ItemCode=@ItemCode and  LotNo=@LotNo and ExpDate is null and MfgDate is
null
        else if @LotNo is null and @ExpDate is not null and @MfgDate is null
                update ItemListLot set LotQty =
                        case
                          when @AdjType=1 or @AdjType=3 then
isnull(LotQty,0)+isnull(@IssueQty,0)
                          when @AdjType=2 or @AdjType=5 then isnull(LotQty,0)-isnull(@IssueQty,0)
                          else isnull(LotQty,0)
                        end
                        ,LotQtyUn =
                        case
                          when @AdjType=4 or @AdjType=5 then
isnull(LotQtyUn,0)+isnull(@IssueQty,0)
                          when @AdjType=6 then isnull(LotQtyUn,0)-isnull(@IssueQty,0)
                          else isnull(LotQtyUn,0)
                        end
                where ItemCode=@ItemCode and  LotNo is null and ExpDate=@ExpDate and MfgDate
is null
        else if @LotNo is null and @ExpDate is null and @MfgDate is not null
                update ItemListLot set LotQty =
                        case
                          when @AdjType=1 or @AdjType=3 then
isnull(LotQty,0)+isnull(@IssueQty,0)
                          when @AdjType=2 or @AdjType=5 then isnull(LotQty,0)-isnull(@IssueQty,0)
                          else isnull(LotQty,0)
                        end
                        ,LotQtyUn =
                        case
```

```
                        when @AdjType=4 or @AdjType=5 then
isnull(LotQtyUn,0)+isnull(@IssueQty,0)
                        when @AdjType=6 then isnull(LotQtyUn,0)-isnull(@IssueQty,0)
                        else isnull(LotQtyUn,0)
                    end
                where ItemCode=@ItemCode and  LotNo is null and ExpDate is null and
MfgDate=@MfgDate
        else if @LotNo is not null and @ExpDate is not null and @MfgDate is null
                update ItemListLot set LotQty =
                    case
                        when @AdjType=1 or @AdjType=3 then
isnull(LotQty,0)+isnull(@IssueQty,0)
                        when @AdjType=2 or @AdjType=5 then isnull(LotQty,0)-isnull(@IssueQty,0)
                        else isnull(LotQty,0)
                    end
                    ,LotQtyUn =
                    case
                        when @AdjType=4 or @AdjType=5 then
isnull(LotQtyUn,0)+isnull(@IssueQty,0)
                        when @AdjType=6 then isnull(LotQtyUn,0)-isnull(@IssueQty,0)
                        else isnull(LotQtyUn,0)
                    end
                where ItemCode=@ItemCode and  LotNo=@LotNo and ExpDate=@ExpDate and
MfgDate is null
        else if @LotNo is null and @ExpDate is not null and @MfgDate is not null
                update ItemListLot set LotQty =
                    case
                        when @AdjType=1 or @AdjType=3 then
isnull(LotQty,0)+isnull(@IssueQty,0)
                        when @AdjType=2 or @AdjType=5 then isnull(LotQty,0)-isnull(@IssueQty,0)
                        else isnull(LotQty,0)
                    end
                    ,LotQtyUn =
                    case
                        when @AdjType=4 or @AdjType=5 then
isnull(LotQtyUn,0)+isnull(@IssueQty,0)
                        when @AdjType=6 then isnull(LotQtyUn,0)-isnull(@IssueQty,0)
                        else isnull(LotQtyUn,0)
                    end
                where ItemCode=@ItemCode and  LotNo is null and ExpDate=@ExpDate and
MfgDate=@MfgDate
        else if @LotNo is not null and @ExpDate is null and @MfgDate is not null
                update ItemListLot set LotQty =
                    case
                        when @AdjType=1 or @AdjType=3 then
isnull(LotQty,0)+isnull(@IssueQty,0)
                        when @AdjType=2 or @AdjType=5 then isnull(LotQty,0)-isnull(@IssueQty,0)
                        else isnull(LotQty,0)
                    end
                    ,LotQtyUn =
                    case
                        when @AdjType=4 or @AdjType=5 then
isnull(LotQtyUn,0)+isnull(@IssueQty,0)
                        when @AdjType=6 then isnull(LotQtyUn,0)-isnull(@IssueQty,0)
                        else isnull(LotQtyUn,0)
                    end
                where ItemCode=@ItemCode and  LotNo=@LotNo and ExpDate is null and
MfgDate=@MfgDate
        else
                update ItemListLot set LotQty =
                    case
```

```
                        when @AdjType=1 or @AdjType=3 then
isnull(LotQty,0)+isnull(@IssueQty,0)
                        when @AdjType=2 or @AdjType=5 then isnull(LotQty,0)-isnull(@IssueQty,0)
                        else isnull(LotQty,0)
                      end
                      ,LotQtyUn =
                      case
                        when @AdjType=4 or @AdjType=5 then
isnull(LotQtyUn,0)+isnull(@IssueQty,0)
                        when @AdjType=6 then isnull(LotQtyUn,0)-isnull(@IssueQty,0)
                        else isnull(LotQtyUn,0)
                      end
              where ItemCode=@ItemCode and  LotNo=@LotNo and ExpDate=@ExpDate and
MfgDate=@MfgDate
        set @tmpCount = @tmpCount + @@ROWCOUNT

        if @tmpCount=1
        begin
                if @AdjType=1 or @AdjType=3
                begin
                        insert into ItemListLot (ItemCode, LotNo, MfgDate,ExpDate,LotQty)
                        values (@ItemCode,@LotNo,@MfgDate,@Expdate,@IssueQty)
                        set @tmpCount = @tmpCount + @@ROWCOUNT
                end
                if @AdjType=4
                begin
                        insert into ItemListLot (ItemCode, LotNo, MfgDate,ExpDate,LotQty,LotQtyUn)
                        values (@ItemCode,@LotNo,@MfgDate,@Expdate,0,@IssueQty)
                        set @tmpCount = @tmpCount + @@ROWCOUNT
                end
        end

        if @tmpCount = 2
                set @UpdCount = @UpdCount + 1
        fetch next from c_AdjInvItems into @ItemCode,@IssueQty,@LotNo,@Mfgdate,@ExpDate
end

close c_AdjInvItems
deallocate c_AdjInvItems

if @UpdCount > 0
begin
        update AdjInv set bUpdated=1 where InvNo=@InvNo
        commit
        set @RecCount = @UpdCount
end

GO
```

PostToStock_IssueInv

This procedure deducts the item quantities of the specific issue invoice from stock. Total number of records updated is returned. Total stock quantity is updated in ItemList table, and lotwise item quantity is updated in ItemListLot table.
Parameters:
@InvType—type of issue invoice: push or indent
@InvNo—adjustment invoice number
@RecCount—total number of records updated
Referenced In: frmIssueInv form, frmIssueIndent form

```
CREATE PROCEDURE dbo.PostToStock_IssueInv
(
	@InvType integer,
	@InvNo integer,
	@RecCount integer output
)
AS

begin transaction

declare @ItemCode varchar(10), @IssueQty decimal, @LotNo varchar(100), @MfgDate datetime,
@ExpDate datetime
declare @tmpCount integer, @UpdCount integer
declare @ItemUsableQty integer

declare c_IndentItems cursor for
select ItemCode, IssueQty, LotNo, MfgDate, ExpDate from IndentItems
where InvType=@InvType and InvNo=@InvNo

open c_IndentItems

set @UpdCount=0
fetch next from c_IndentItems into @ItemCode,@IssueQty,@LotNo,@Mfgdate,@ExpDate
while @@FETCH_STATUS=0
begin
	set @tmpCount=0

	select @ItemUsableQty = UsableQty from ItemList where ItemCode=@ItemCode
	if isnull(@ItemUsableQty,0)-isnull(@IssueQty,0) >= 0
	  begin
		update ItemList set UsableQty = isnull(UsableQty,0)-isnull(@IssueQty,0)
		 where ItemCode=@ItemCode
		set @tmpCount = @@ROWCOUNT
	   end
	else
	   begin
		rollback
		set @RecCount = -1
		return
	   end

	if @LotNo is null and @ExpDate is null and @MfgDate is null
		update ItemListLot set LotQty = isnull(LotQty,0)-isnull(@IssueQty,0)
		where ItemCode=@ItemCode and LotNo is null and ExpDate is null and MfgDate is null
	else if  @LotNo is not null and @ExpDate is null and @MfgDate is null
		update ItemListLot set LotQty = isnull(LotQty,0)-isnull(@IssueQty,0)
		where ItemCode=@ItemCode and  LotNo=@LotNo and ExpDate is null and MfgDate is
null
	else if @LotNo is null and @ExpDate is not null and @MfgDate is null
		update ItemListLot set LotQty = isnull(LotQty,0)-isnull(@IssueQty,0)
		where ItemCode=@ItemCode and  LotNo is null and ExpDate=@ExpDate and MfgDate
is null
	else if @LotNo is null and @ExpDate is null and @MfgDate is not null
		update ItemListLot set LotQty = isnull(LotQty,0)-isnull(@IssueQty,0)
		where ItemCode=@ItemCode and  LotNo is null and ExpDate is null and
MfgDate=@MfgDate
	else if @LotNo is not null and @ExpDate is not null and @MfgDate is null
		update ItemListLot set LotQty = isnull(LotQty,0)-isnull(@IssueQty,0)
		where ItemCode=@ItemCode and  LotNo=@LotNo and ExpDate=@ExpDate and
MfgDate is null
	else if @LotNo is null and @ExpDate is not null and @MfgDate is not null
```

```
                update ItemListLot set LotQty = isnull(LotQty,0)-isnull(@IssueQty,0)
                where ItemCode=@ItemCode and  LotNo is null and ExpDate=@ExpDate and
MfgDate=@MfgDate
        else if @LotNo is not null and @ExpDate is null and @MfgDate is not null
                update ItemListLot set LotQty = isnull(LotQty,0)-isnull(@IssueQty,0)
                where ItemCode=@ItemCode and  LotNo=@LotNo and ExpDate is null and
MfgDate=@MfgDate
        else
                update ItemListLot set LotQty = isnull(LotQty,0)-isnull(@IssueQty,0)
                where ItemCode=@ItemCode and  LotNo=@LotNo and ExpDate=@ExpDate and
MfgDate=@MfgDate
        set @tmpCount = @tmpCount + @@ROWCOUNT

        if @tmpCount = 2
                set @UpdCount = @UpdCount + 1
        fetch next from c_IndentItems into @ItemCode,@IssueQty,@LotNo,@Mfgdate,@ExpDate
end

close c_IndentItems
deallocate c_IndentItems

if @UpdCount > 0
begin
        update Indent set bUpdated=1 where InvType=@InvType and InvNo=@InvNo
        commit
        set @RecCount = @UpdCount
end

GO
```

PostToStock_RecInv

This procedure adds up the item quantities from stock of the specific receive invoice. Total number of records updated is returned. Yotal stock quantity is updated in ItemList table, and lotwise item quantity is updated in ItemListLot table.
Parameters:
@InvType—type of receive invoice: receive from external supplier or from warehouse
@InvNo—adjustment invoice number
@RecCount—total number of records updated
Referenced In: frmRecInv form, frmRecInvFac form

```
CREATE PROCEDURE dbo.PostToStock_RecInv
(
        @InvType integer,
        @InvNo integer,
        @RecCount integer output
)
AS

begin transaction

declare @ItemCode varchar(10), @Qty decimal, @LotNo varchar(100), @MfgDate datetime,
@ExpDate datetime
declare @tmpCount integer, @UpdCount integer

declare c_RecInvItems cursor for
select ItemCode, Qty, LotNo, MfgDate, ExpDate from RecInvItems
where InvType=@InvType and InvNo=@InvNo

open c_RecInvItems
```

```
set @UpdCount=0
fetch next from c_RecInvItems into @ItemCode,@Qty,@LotNo,@Mfgdate,@ExpDate
while @@FETCH_STATUS=0
begin
	set @tmpCount=0

	update ItemList set UsableQty = isnull(UsableQty,0)+isnull(@Qty,0)
	 where ItemCode=@ItemCode
	set @tmpCount = @@ROWCOUNT

	if @LotNo is null and @ExpDate is null and @MfgDate is null
		update ItemListLot set LotQty = isnull(LotQty,0)+isnull(@Qty,0)
		where ItemCode=@ItemCode and LotNo is null and ExpDate is null and MfgDate is null
	else if  @LotNo is not null and @ExpDate is null and @MfgDate is null
		update ItemListLot set LotQty = isnull(LotQty,0)+isnull(@Qty,0)
		where ItemCode=@ItemCode and  LotNo=@LotNo and ExpDate is null and MfgDate is
null
	else if @LotNo is null and @ExpDate is not null and @MfgDate is null
		update ItemListLot set LotQty = isnull(LotQty,0)+isnull(@Qty,0)
		where ItemCode=@ItemCode and  LotNo is null and ExpDate=@ExpDate and MfgDate
is null
	else if @LotNo is null and @ExpDate is null and @MfgDate is not null
		update ItemListLot set LotQty = isnull(LotQty,0)+isnull(@Qty,0)
		where ItemCode=@ItemCode and  LotNo is null and ExpDate is null and
MfgDate=@MfgDate
	else if  @LotNo is not null and @ExpDate is not null and @MfgDate is null
		update ItemListLot set LotQty = isnull(LotQty,0)+isnull(@Qty,0)
		where ItemCode=@ItemCode and  LotNo=@LotNo and ExpDate=@ExpDate and
MfgDate is null
	else if @LotNo is null and @ExpDate is not null and @MfgDate is not null
		update ItemListLot set LotQty = isnull(LotQty,0)+isnull(@Qty,0)
		where ItemCode=@ItemCode and  LotNo is null and ExpDate=@ExpDate and
MfgDate=@MfgDate
	else if @LotNo is not null and @ExpDate is null and @MfgDate is not null
		update ItemListLot set LotQty = isnull(LotQty,0)+isnull(@Qty,0)
		where ItemCode=@ItemCode and  LotNo=@LotNo and ExpDate is null and
MfgDate=@MfgDate
	else
		update ItemListLot set LotQty = isnull(LotQty,0)+isnull(@Qty,0)
		where ItemCode=@ItemCode and  LotNo=@LotNo and ExpDate=@ExpDate and
MfgDate=@MfgDate
	set @tmpCount = @tmpCount + @@ROWCOUNT

	if @tmpCount=1
	begin
		insert into ItemListLot (ItemCode, LotNo, MfgDate,ExpDate,LotQty)
		values (@ItemCode,@LotNo,@MfgDate,@Expdate,@Qty)
		set @tmpCount = @tmpCount + @@ROWCOUNT
	end

	if @tmpCount = 2
		set @UpdCount = @UpdCount + 1
	fetch next from c_RecInvItems into @ItemCode,@Qty,@LotNo,@Mfgdate,@ExpDate
end

close c_RecInvItems
deallocate c_RecInvItems

if @UpdCount > 0
begin
```

```
        update RecInv set bUpdated=1 where InvType=@InvType and InvNo=@InvNo
        commit
        set @RecCount = @UpdCount
end

GO
```

WIMS Menu Structure

- File
  - Exit
- Parameters
  - Item Groups...
  - Item List...
  - Suppliers...
- Invoice
  - Receive from Supplier...
  - Transfer/Receive from Warehouse...
  - Issue Voucher (Indent)...
  - Issue Voucher (Push)...
  - Gate Pass...
  - Adjustment Invoice...
- Stock
  - Stock Balance...
  - Stock Reports...
- Administration
  - Login...
  - Logout...
  - Change Password...
  - Employees...
  - Districts...
  - Facility List...
  - Lookup Tables...
  - Backup Database...
  - Options...
- Window
- Help
  - Contents...
  - About WIMS...

Reports

AdjInv.rpt

This report prints any Adjustment Invoice.
Stored Procedure Name: GetAdjInvRpt
Parameters:
@InvNo—Adjustment Invoice number to print
Tables Used: AdjInv, AdjInvItems, ItemList, Facility, UnitOfMeas, Employee
Referenced In: frmAdjInv form

```
CREATE PROCEDURE dbo.GetAdjInvRpt
(
        @InvNo integer
)
 AS

SELECT AdjInv.InvNo, AdjInv.FacilityCode, Facility.FacilityName, AdjInv.bUpdated,
AdjInv.AdjRemarks, AdjInvItems.ItemSl,
AdjInvItems.ItemCode, ItemList.ItemName, UnitOfMeas.UnitName, AdjInvItems.IssueQty,
AdjInvItems.ExpDate,
(select EmpName from Employee where EmpCode=AdjInv.PrepBy) AS NamePrepBy,
(select Designation from Designation inner join Employee on
Designation.DesigCode=Employee.DesigCode where EmpCode=AdjInv.PrepBy) AS DesigPrepBy,
AdjInv.PrepDate,
(select EmpName from Employee where EmpCode=AdjInv.AppBy1) AS NameAppBy1,
(select Designation from Designation inner join Employee on
Designation.DesigCode=Employee.DesigCode where EmpCode=AdjInv.AppBy1) AS DesigAppBy1,
AdjInv.AppDate1,
(select EmpName from Employee where EmpCode=AdjInv.IssuedBy) AS NameIssuedBy,
(select Designation from Designation inner join Employee on
Designation.DesigCode=Employee.DesigCode where EmpCode=AdjInv.issuedBy) AS DesigIssuedBy,
AdjInv.IssuedDate, AdjInvItems.LotNo,
(select FacilityName from Facility where FacilityCode=AdjInv.fromFacility) AS FromFacilityName
FROM UnitOfMeas INNER JOIN (ItemList INNER JOIN ((Facility INNER JOIN AdjInv ON
Facility.FacilityCode = AdjInv.FacilityCode) INNER JOIN AdjInvItems ON (AdjInv.InvNo =
AdjInvItems.InvNo))
ON ItemList.ItemCode = AdjInvItems.ItemCode) ON UnitOfMeas.UnitId = ItemList.UnitId
WHERE AdjInv.InvNo=@InvNo

GO
```

Districts.rpt

Prints the list of districts in Bangladesh.
Stored Procedure Name: None (based on ad hoc query)
Parameters: None
Tables Used: District, Division
Referenced In: frmDistrict form

```
SELECT
   Division.DivName,
   District.DistCode, District.DistName
FROM
   Division INNER JOIN District ON Division.DivCode = District.DivCode
```

```
WHERE
   Division.DivCode = District.DivCode
ORDER BY
   Division.DivName, District.DistName;
```

FacilityList.rpt

Prints the list of facilities in the system, which are sorted according to their group and then by name.
Stored Procedure Name: GetFacilityListRpt
Parameters: None
Tables Used: FGroup, Facility, District
Referenced In: frmFacility form

```
CREATE PROCEDURE dbo.GetFacilityListRpt AS

SELECT [FGroup].[FGroupName], [District].[DistName], [Facility].[FacilityCode],
[Facility].[FacilityName]
FROM FGroup INNER JOIN (District INNER JOIN Facility ON [District].[DistCode]=[Facility].[DistCode])
ON [FGroup].[FGroupCode]=left([Facility].[FacilityCode],1)
Order by [FGroup].[FGroupName], [Facility].[FacilityName]

GO
```

FacilityListByDist.rpt

Prints the list of facilities in the system, which are grouped based on district and then by name.
Stored Procedure Name: GetFacilityListRpt
Parameters: None
Tables Used: FGroup, Facility, District
Referenced In: frmFacility form

```
CREATE PROCEDURE dbo.GetFacilityListRpt AS

SELECT [FGroup].[FGroupName], [District].[DistName], [Facility].[FacilityCode],
[Facility].[FacilityName]
FROM FGroup INNER JOIN (District INNER JOIN Facility ON [District].[DistCode]=[Facility].[DistCode])
ON [FGroup].[FGroupCode]=left([Facility].[FacilityCode],1)
Order by [FGroup].[FGroupName], [Facility].[FacilityName]

GO
```

GatePass.rpt

Prints the master part of a Gate Pass.
Stored Procedure Name: GetGatePassRpt
Parameter:
@GPassNo—Gate Pass number to print
Tables Used: GatePass, GatePassItems, Indent, Facility
Referenced In: frmGatePass form

```
CREATE PROCEDURE dbo.GetGatePassRpt
(
        @GPassNo integer
)
 AS
```

```
SELECT GatePass.GPassNo, GPassDate, DriverName, VehicleNo, GatePassItems.InvNo, FacilityName,
GatePassItems.InvType
FROM GatePass INNER JOIN (GatePassItems INNER JOIN (Indent INNER JOIN Facility ON
Indent.FacilityCode=facility.FacilityCode) ON GatePassItems.InvType=Indent.InvType and
GatePassItems.InvNo=Indent.InvNo)
ON GatePass.GPassNo=GatePassItems.GPassNo
WHERE GatePass.GPassNo=@GPassNo

GO
```

GatePassItems.rpt

This subreport prints the details part of a Gate Pass.
Stored Procedure Name: GetGatePassItemsRpt
Parameters: None
Tables Used: GatePassItems, Indent, Facility
Referenced In: frmGatePass form

```
CREATE PROCEDURE dbo.GetGatePassItemsRpt
 AS

SELECT GPassNo, GatePassItems.InvNo, FacilityName, GatePassItems.InvType
FROM GatePassItems INNER JOIN (Indent INNER JOIN Facility ON
Indent.FacilityCode=facility.FacilityCode) ON GatePassItems.InvType=Indent.InvType and
GatePassItems.InvNo=Indent.InvNo

GO
```

IndentInv.rpt, IssueInv.rpt

Both of these reports print any issue invoice. IndentInv prints only invoices based on indent, and IssueInv prints push issue invoices.
Stored Procedure Name: GetIssueInvRpt
Parameters:
@InvType—type of issue invoice; 1: push, 2: indent
@InvNo—Invoice no. to print
Tables Used: Indent, IndentItems, Facility, UnitOfMeas, ItemList, GatePass, GatePassItems, Warehouse, Employee
Referenced In: frmIssueIndent form, frmIssueInv form

```
CREATE PROCEDURE dbo.GetIssueInvRpt
(
        @InvType integer,
        @InvNo integer
)
 AS

SELECT Indent.InvType, Indent.InvNo, Indent.IndentNo, Indent.IndentPrepDate,
(select FacilityName from Warehouse) as FromFacility,
Indent.FacilityCode, Facility.FacilityName, Indent.bUpdated, IndentItems.ItemSl,
IndentItems.ItemCode, ItemList.ItemName, UnitOfMeas.UnitName, IndentItems.ReqQty,
IndentItems.IssueQty, IndentItems.ExpDate,
(select EmpName from Employee where EmpCode=Indent.PrepBy) AS NamePrepBy,
(select Designation from Designation inner join Employee on
Designation.DesigCode=Employee.DesigCode where EmpCode=Indent.PrepBy) AS DesigPrepBy,
Indent.PrepDate,
```

```
(select EmpName from Employee where EmpCode=Indent.AppBy1) AS NameAppBy1,
(select Designation from Designation inner join Employee on
Designation.DesigCode=Employee.DesigCode where EmpCode=Indent.AppBy1) AS DesigAppBy1,
Indent.AppDate1,
(select EmpName from Employee where EmpCode=Indent.IssuedBy) AS NameIssuedBy,
(select Designation from Designation inner join Employee on
Designation.DesigCode=Employee.DesigCode where EmpCode=Indent.issuedBy) AS DesigIssuedBy,
Indent.IssuedDate,
DriverName, VehicleNo, Indent.Remarks, IndentItems.LotNo
FROM ((Facility INNER JOIN Indent ON Facility.FacilityCode = Indent.FacilityCode) INNER JOIN
(UnitOfMeas INNER JOIN (ItemList INNER JOIN IndentItems ON ItemList.ItemCode =
IndentItems.ItemCode) ON
UnitOfMeas.UnitId = ItemList.UnitId) ON (Indent.InvType = IndentItems.InvType) AND
(Indent.InvNo = IndentItems.InvNo)) LEFT JOIN (GatePass INNER JOIN GatePassItems ON
GatePass.GPassNo = GatePassItems.GPassNo) ON (Indent.InvType = GatePassItems.InvType) AND
(Indent.InvNo = GatePassItems.InvNo)
WHERE Indent.InvType=@InvType and Indent.InvNo=@InvNo
order by IndentItems.ItemSl

GO
```

ItemGroups.rpt

Prints the list of Item Groups in the database.
Stored Procedure Name: None (based on ItemGroup Table)
Parameters: None
Tables Used: ItemGroup
Referenced In: frmItemGroup form

RecInv.rpt

Prints receive invoices that receive commodities from external supplier.
Stored Procedure Name: GetRecInvRpt
Parameters:
@InvType—type of receive invoice; 1: receive from external source, 2: internal transfer from warehouse to warehouse
@InvNo—invoice number to print
Tables Used: RecInv, RecInvItems, Supplier, UnitOfMeas, Employee, ItemList
Referenced In: frmRecInv form

```
CREATE PROCEDURE dbo.GetRecInvRpt
(
        @InvType integer,
        @InvNo integer
)
 AS

SELECT RecInv.InvType, RecInv.InvNo, RecInv.ReceiptDate, Employee.EmpName,
Supplier.SupName, RecInv.SupRef, RecInv.SupDate, RecInv.bUpdated,
RecInvItems.ItemSl, RecInvItems.ItemCode, ItemList.ItemName,
UnitOfMeas.UnitName, RecInvItems.Qty, RecInvItems.UnitPrice, RecInvItems.ExpDate,
RecInvItems.MfgDate, RecInvItems.LotNo, RecInv.BLNo, RecInv.BLDate
FROM UnitOfMeas INNER JOIN (Supplier INNER JOIN ((Employee INNER JOIN RecInv
ON Employee.EmpCode = RecInv.ReceivedBy) INNER JOIN (ItemList INNER JOIN
RecInvItems ON ItemList.ItemCode = RecInvItems.ItemCode) ON
RecInv.InvType = RecInvItems.InvType and RecInv.InvNo = RecInvItems.InvNo) ON
Supplier.SupCode = RecInv.SupCode)
```

```
ON UnitOfMeas.UnitId = ItemList.UnitId
WHERE RecInv.InvType=@InvType and RecInv.InvNo=@InvNo

GO
```

RecInvFac.rpt

Prints receive invoices that transfer commodities from one warehouse to another.
Stored Procedure Name: GetRecInvFacRpt
Parameters:
@InvType—type of receive invoice; 1: receive from external source, 2: internal transfer from warehouse to warehouse
@InvNo—invoice number to print
Tables Used: RecInv, RecInvItems, UnitOfMeas, Employee, ItemList, Facility
Referenced In: frmrecInvfac form

```
CREATE PROCEDURE dbo.GetRecInvFacRpt
(
        @InvType integer,
        @InvNo integer
)
 AS

SELECT RecInv.InvType, RecInv.InvNo, RecInv.ReceiptDate, Employee.EmpName,
Facility.FacilityName, RecInv.SupRef, RecInv.SupDate, RecInv.bUpdated,
RecInvItems.ItemSl, RecInvItems.ItemCode, ItemList.ItemName,
UnitOfMeas.UnitName, RecInvItems.Qty, RecInvItems.UnitPrice, RecInvItems.ExpDate,
RecInvItems.MfgDate, RecInvItems.LotNo, RecInv.BLNo, RecInv.BLDate
FROM UnitOfMeas INNER JOIN (Facility INNER JOIN ((Employee INNER JOIN RecInv
ON Employee.EmpCode = RecInv.ReceivedBy) INNER JOIN (ItemList INNER JOIN
RecInvItems ON ItemList.ItemCode = RecInvItems.ItemCode) ON
RecInv.InvType = RecInvItems.InvType and RecInv.InvNo = RecInvItems.InvNo) ON
Facility.FacilityCode = RecInv.FromFacility)
ON UnitOfMeas.UnitId = ItemList.UnitId
WHERE RecInv.InvType=@InvType and RecInv.InvNo=@InvNo

GO
```

StockDetails.rpt

Prints detailed transactions of a specific item. Parameters are selected from a selection form, which computes and pre-fills tmpStockDetails table. This table data are used to display the report.
Stored Procedure Name: None (tmpStockDetails Table)
Parameters: None
Tables Used: tmpStockDetails Table
Referenced In: frmStockDetails form, frmReports form

StockDetails_byFacility.rpt

Prints detailed transactions for a specific facility. Parameters are selected from a selection form that computes and pre-fills tmpStockDetails table. This table data are used to illustrate the report.
Stored Procedure Name: None (tmpStockDetails Table)
Parameters: None
Tables Used: tmpStockDetails Table
Referenced In: frmStockDetails form, frmReports form

StockSummary.rpt

Prints the summary stock of commodities as of the current date.
Stored Procedure Name: GetStockSummaryRpt
Parameters:
@pGroup—specific Item Group for which stock summary will be printed, i.e., CON, IEM. An empty string is passed to print stock summary of all groups.
@pNonZero—0: prints only non-zero items, 1: prints only zero items, 2: prints both zero and non-zero items (all items)
Tables Used: UnitOfMeas, ItemGroup, ItemList
Referenced In: frmStockSummary form, frmReports form

```
CREATE PROCEDURE dbo.GetStockSummaryRpt
(
        @pGroup varchar(10),
        @pNonZero integer
)
 AS

if isnull(@pGroup,'') <> ''
   begin
        if @pNonZero=2
                SELECT ItemGroup.GroupCode, ItemGroup.GroupName, ItemList.ItemCode,
ItemList.ItemName,
                ItemList.UsableQty, ItemList.UnusableQty, UnitOfMeas.UnitName
                FROM UnitOfMeas RIGHT JOIN (ItemGroup INNER JOIN ItemList ON
ItemGroup.GroupCode =
                ItemList.GroupCode) ON UnitOfMeas.UnitId = ItemList.UnitId
                where ItemGroup.GroupCode=@pGroup
        else if @pNonZero=0
                SELECT ItemGroup.GroupCode, ItemGroup.GroupName, ItemList.ItemCode,
ItemList.ItemName,
                ItemList.UsableQty, ItemList.UnusableQty, UnitOfMeas.UnitName
                FROM UnitOfMeas RIGHT JOIN (ItemGroup INNER JOIN ItemList ON
ItemGroup.GroupCode =
                ItemList.GroupCode) ON UnitOfMeas.UnitId = ItemList.UnitId
                where ItemGroup.GroupCode=@pGroup and
                (ItemList.UsableQty > 0 or ItemList.UnusableQty>0)
        else if @pNonZero=1
                SELECT ItemGroup.GroupCode, ItemGroup.GroupName, ItemList.ItemCode,
ItemList.ItemName,
                ItemList.UsableQty, ItemList.UnusableQty, UnitOfMeas.UnitName
                FROM UnitOfMeas RIGHT JOIN (ItemGroup INNER JOIN ItemList ON
ItemGroup.GroupCode =
                ItemList.GroupCode) ON UnitOfMeas.UnitId = ItemList.UnitId
                where ItemGroup.GroupCode=@pGroup and
                (ItemList.UsableQty = 0 and ItemList.UnusableQty = 0)
   end
else
   begin
        if @pNonZero=2
                SELECT ItemGroup.GroupCode, ItemGroup.GroupName, ItemList.ItemCode,
ItemList.ItemName,
                ItemList.UsableQty, ItemList.UnusableQty, UnitOfMeas.UnitName
                FROM UnitOfMeas RIGHT JOIN (ItemGroup INNER JOIN ItemList ON
ItemGroup.GroupCode =
                ItemList.GroupCode) ON UnitOfMeas.UnitId = ItemList.UnitId
        else if @pNonZero=0
```

```
                SELECT ItemGroup.GroupCode, ItemGroup.GroupName, ItemList.ItemCode,
ItemList.ItemName,
                ItemList.UsableQty, ItemList.UnusableQty, UnitOfMeas.UnitName
                FROM UnitOfMeas RIGHT JOIN (ItemGroup INNER JOIN ItemList ON
ItemGroup.GroupCode =
                ItemList.GroupCode) ON UnitOfMeas.UnitId = ItemList.UnitId
                where  (ItemList.UsableQty > 0 or ItemList.UnusableQty>0)
        else if @pNonZero=1
                SELECT ItemGroup.GroupCode, ItemGroup.GroupName, ItemList.ItemCode,
ItemList.ItemName,
                ItemList.UsableQty, ItemList.UnusableQty, UnitOfMeas.UnitName
                FROM UnitOfMeas RIGHT JOIN (ItemGroup INNER JOIN ItemList ON
ItemGroup.GroupCode =
                ItemList.GroupCode) ON UnitOfMeas.UnitId = ItemList.UnitId
                where  (ItemList.UsableQty = 0 and ItemList.UnusableQty = 0)
   end

GO
```

StockSummaryAny.rpt

Prints the summary stock of commodities as of a previous date.
Stored Procedure Name: GetStockSummaryAnyRpt
Parameters:
@pGroup—specific Item Group for which stock summary will be printed, i.e., CON, IEM. An empty string is passed to print stock summary of all groups.
@pNonZero—0: prints only non-zero items, 1: prints only zero items, 2: prints both zero and non-zero items (all items)
@pBalDate—date of stock summary
Tables Used: UnitOfMeas, ItemGroup, ItemList, RecInv, RecInvItems, Indent, IndentItems, AdjType, AdjInv, AdjInvItems
Referenced In: frmStockSummary form, frmReports form

```
CREATE PROCEDURE dbo.GetStockSummaryAnyRpt
(
        @pGroup varchar(10),
        @pNonZero integer,
        @pBalDate varchar(20)
)
 AS

declare @StrSql nvarchar(2000)
declare @Criteria nvarchar(500)

set @StrSql =
'select ItemList.GroupCode, ItemGroup.GroupName, ItemList.ItemCode, ItemList.ItemName,
sum(QtyIn-QtyOut) as UsableQty, sum(UnQtyIn-UnQtyOut) as UnusableQty, UnitName ' +
'from UnitOfMeas inner join (ItemGroup inner join (ItemList inner join (' +
'select ItemCode, InitStock as QtyIn, 0 as QtyOut, InitStockUn as UnQtyIn, 0 as UnQtyOut ' +
'from ItemList  ' +
'UNION ALL ' +
'select ItemCode, sum(Qty) as QtyIn, 0 as QtyOut, 0 as UnQtyIn, 0 as UnQtyOut ' +
'from RecInv INNER JOIN RecInvItems ON (RecInv.InvNo = RecInvItems.InvNo) AND (RecInv.InvType
= RecInvItems.InvType)  ' +
'where bUpdated=1 and ReceiptDate <= ' + char(39) +  @pBalDate + char(39) + ' group by
ItemCode  UNION ALL ' +
'select ItemCode, 0 as QtyIn, sum(IssueQty) as QtyOut, 0 as UnQtyIn, 0 as UnQtyOut ' +
'from Indent INNER JOIN IndentItems ON (Indent.InvNo = IndentItems.InvNo) AND (Indent.InvType
= IndentItems.InvType)  ' +
```

```
'where  bUpdated=1 and IssuedDate <= ' + char(39) + @pBalDate + char(39) + ' and IssueQty is not
NULL group by ItemCode  UNION ALL ' +
'select ItemCode, sum(IssueQty*usablein) as QtyIn, sum(IssueQty*usableout) as QtyOut,
sum(IssueQty*unusablein) as UnQtyIn, sum(IssueQty*unusableout) as UnQtyOut ' +
'from AdjType INNER JOIN (AdjInv INNER JOIN AdjInvItems ON (AdjInv.InvNo = AdjInvItems.InvNo))
ON AdjType.AdjType=AdjInv.AdjType  ' +
'where  bUpdated=1 and IssuedDate <= ' + char(39) + @pBalDate + char(39) + ' group by ItemCode
' +
') a  on a.ItemCode = ItemList.ItemCode) on ItemGroup.GroupCode = ItemList.GroupCode) on
UnitOfMeas.UnitId = ItemList.UnitId '

if isnull(@pGroup,'') <> ''
        set @StrSql = @StrSql +  ' where ItemList.GroupCode=' + char(39)  + @pGroup + char(39)
set @StrSql = @StrSql + ' group by ItemList.GroupCode, ItemGroup.GroupName, ItemList.ItemName,
UnitName, ItemList.ItemCode '

if @pNonZero=0
        set @StrSql = @StrSql + ' having sum(QtyIn-QtyOut)>0 '
else if @pNonZero=1
        set @StrSql = @StrSql + ' having sum(QtyIn-QtyOut)=0 '
else
        set @StrSql = @StrSql

exec sp_executesql @StrSql

GO
```

StockSummaryLot.rpt

Prints the lotwise summary stock of commodities as of the current date.
Stored Procedure Name: GetStockSummaryLotRpt
Parameters:
@pGroup—specific Item Group for which stock summary will be printed, i.e., CON, IEM. An empty string is passed to print stock summary of all groups.
@pNonZero—0: prints only non-zero items, 1: prints only zero items, 2: prints both zero and non-zero items (all items)
Tables Used: UnitOfMeas, ItemGroup, ItemList, ItemListLot
Referenced In: frmReports form

```
CREATE PROCEDURE dbo.GetStockSummaryLotRpt
(
        @pGroup varchar(10),
        @pNonZero integer
)
 AS

if isnull(@pGroup,'') <> ''
   begin
        if @pNonZero=2
                SELECT ItemGroup.GroupCode, ItemGroup.GroupName, ItemList.ItemCode,
ItemList.ItemName,
                LotNo, MfgDate, ExpDate, LotQty, LotQtyUn, UnitOfMeas.UnitName
                FROM UnitOfMeas RIGHT JOIN (ItemGroup INNER JOIN
                (ItemList INNER JOIN ItemListLot ON ItemList.ItemCode=ItemListLot.ItemCode) ON
ItemGroup.GroupCode =
                ItemList.GroupCode) ON UnitOfMeas.UnitId = ItemList.UnitId
                where ItemGroup.GroupCode=@pGroup
        else if @pNonZero=0
                SELECT ItemGroup.GroupCode, ItemGroup.GroupName, ItemList.ItemCode,
ItemList.ItemName,
```

```
                LotNo, MfgDate, ExpDate, LotQty, LotQtyUn, UnitOfMeas.UnitName
                FROM UnitOfMeas RIGHT JOIN (ItemGroup INNER JOIN
                (ItemList INNER JOIN ItemListLot ON ItemList.ItemCode=ItemListLot.ItemCode) ON
ItemGroup.GroupCode =
                ItemList.GroupCode) ON UnitOfMeas.UnitId = ItemList.UnitId
                where ItemGroup.GroupCode=@pGroup and
                (ItemListLot.LotQty > 0 or ItemListLot.LotQtyUn>0)
        else if @pNonZero=1
                SELECT ItemGroup.GroupCode, ItemGroup.GroupName, ItemList.ItemCode,
ItemList.ItemName,
                LotNo, MfgDate, ExpDate, LotQty, LotQtyUn, UnitOfMeas.UnitName
                FROM UnitOfMeas RIGHT JOIN (ItemGroup INNER JOIN
                (ItemList INNER JOIN ItemListLot ON ItemList.ItemCode=ItemListLot.ItemCode) ON
ItemGroup.GroupCode =
                ItemList.GroupCode) ON UnitOfMeas.UnitId = ItemList.UnitId
                where ItemGroup.GroupCode=@pGroup and
                (ItemListLot.LotQty = 0 and ItemListLot.LotQtyUn = 0)
   end
else
   begin
        if @pNonZero=2
                SELECT ItemGroup.GroupCode, ItemGroup.GroupName, ItemList.ItemCode,
ItemList.ItemName,
                LotNo, MfgDate, ExpDate, LotQty, LotQtyUn, UnitOfMeas.UnitName
                FROM UnitOfMeas RIGHT JOIN (ItemGroup INNER JOIN
                (ItemList INNER JOIN ItemListLot ON ItemList.ItemCode=ItemListLot.ItemCode) ON
ItemGroup.GroupCode =
                ItemList.GroupCode) ON UnitOfMeas.UnitId = ItemList.UnitId
        else if @pNonZero=0
                SELECT ItemGroup.GroupCode, ItemGroup.GroupName, ItemList.ItemCode,
ItemList.ItemName,
                LotNo, MfgDate, ExpDate, LotQty, LotQtyUn, UnitOfMeas.UnitName
                FROM UnitOfMeas RIGHT JOIN (ItemGroup INNER JOIN
                (ItemList INNER JOIN ItemListLot ON ItemList.ItemCode=ItemListLot.ItemCode) ON
ItemGroup.GroupCode =
                ItemList.GroupCode) ON UnitOfMeas.UnitId = ItemList.UnitId
                where (ItemListLot.LotQty > 0 or ItemListLot.LotQtyUn>0)
        else if @pNonZero=1
                SELECT ItemGroup.GroupCode, ItemGroup.GroupName, ItemList.ItemCode,
ItemList.ItemName,
                LotNo, MfgDate, ExpDate, LotQty, LotQtyUn, UnitOfMeas.UnitName
                FROM UnitOfMeas RIGHT JOIN (ItemGroup INNER JOIN
                (ItemList INNER JOIN ItemListLot ON ItemList.ItemCode=ItemListLot.ItemCode) ON
ItemGroup.GroupCode =
                ItemList.GroupCode) ON UnitOfMeas.UnitId = ItemList.UnitId
                where (ItemListLot.LotQty = 0 and ItemListLot.LotQtyUn = 0)
   end

GO
```

StockSummaryLotAny.rpt

Prints the lotwise summary stock of commodities as of a previous date.
Stored Procedure Name: GetStockSummaryLotAnyRpt
Parameters:
@pGroup—specific Item Group for which stock summary will be printed, i.e., CON, IEM. An empty string is passed to print stock summary of all groups.
@pNonZero—0" prints only non-zero items, 1: prints only zero items, 2: prints both zero and non-zero items (all items)
@pBalDate—date of stock summary

Tables Used: UnitOfMeas, ItemGroup, ItemList, RecInv, RecInvItems, Indent, IndentItems, AdjType, AdjInv, AdjInvItems, ItemListLot
Referenced In: frmReports form

```
CREATE PROCEDURE dbo.GetStockSummaryLotAnyRpt
(
	@pGroup varchar(10),
	@pNonZero integer,
	@pBalDate varchar(20)
)
 AS

declare @StrSql nvarchar(2000)
declare @Criteria nvarchar(500)

set @StrSql =
'select ItemList.GroupCode, ItemGroup.GroupName, ItemList.ItemCode, ItemList.ItemName,
LotNo, MfgDate, ExpDate,
sum(QtyIn-QtyOut) as UsableQty, sum(UnQtyIn-UnQtyOut) as UnusableQty, UnitName ' +
'from UnitOfMeas inner join (ItemGroup inner join (ItemList inner join (' +
'select ItemCode, LotNo, MfgDate, ExpDate,  InitLotQty as QtyIn, 0 as QtyOut, InitLotQtyUn as
UnQtyIn, 0 as UnQtyOut ' +
'from ItemListLot  ' +
'UNION ALL ' +
'select ItemCode, LotNo, MfgDate, ExpDate, sum(Qty) as QtyIn, 0 as QtyOut, 0 as UnQtyIn, 0 as
UnQtyOut ' +
'from RecInv INNER JOIN RecInvItems ON (RecInv.InvNo = RecInvItems.InvNo) AND (RecInv.InvType
= RecInvItems.InvType)  ' +
'where bUpdated=1 and ReceiptDate <= ' + char(39) +  @pBalDate + char(39) + ' group by
ItemCode, LotNo, MfgDate, ExpDate  UNION ALL ' +
'select ItemCode, LotNo, MfgDate, ExpDate,  0 as QtyIn, sum(IssueQty) as QtyOut, 0 as UnQtyIn, 0
as UnQtyOut ' +
'from Indent INNER JOIN IndentItems ON (Indent.InvNo = IndentItems.InvNo) AND (Indent.InvType
= IndentItems.InvType)  ' +
'where  bUpdated=1 and IssuedDate <= ' + char(39) + @pBalDate + char(39) + ' and IssueQty is not
NULL group by ItemCode, LotNo, MfgDate, ExpDate  UNION ALL ' +
'select ItemCode, LotNo, MfgDate, ExpDate,  sum(IssueQty*usablein) as QtyIn,
sum(IssueQty*usableout) as QtyOut, sum(IssueQty*unusablein) as UnQtyIn,
sum(IssueQty*unusableout) as UnQtyOut ' +
'from AdjType INNER JOIN (AdjInv INNER JOIN AdjInvItems ON (AdjInv.InvNo = AdjInvItems.InvNo))
ON AdjType.AdjType=AdjInv.AdjType  ' +
'where  bUpdated=1 and IssuedDate <= ' + char(39) + @pBalDate + char(39) + ' group by
ItemCode, LotNo, MfgDate, ExpDate ' +
') a  on a.ItemCode = ItemList.ItemCode) on ItemGroup.GroupCode = ItemList.GroupCode) on
UnitOfMeas.UnitId = ItemList.UnitId '

if isnull(@pGroup,'') <> ''
	set @StrSql = @StrSql +  ' where ItemList.GroupCode=' + char(39)  + @pGroup + char(39)
set @StrSql = @StrSql + ' group by ItemList.GroupCode, ItemGroup.GroupName, ItemList.ItemName,
UnitName, ItemList.ItemCode,LotNo, MfgDate, ExpDate '

if @pNonZero=0
	set @StrSql = @StrSql + ' having sum(QtyIn-QtyOut)>0 '
else if @pNonZero=1
	set @StrSql = @StrSql + ' having sum(QtyIn-QtyOut)=0 '
else
	set @StrSql = @StrSql

exec sp_executesql @StrSql

GO
```

Suppliers.rpt

Prints the list of external suppliers of commodities.
Stored Procedure Name: None (Supplier Table)
Parameters: None
Tables Used: Supplier Table
Referenced In: frmSupplier form

Data Entry Forms

There are 25 data entry forms in WIMS, all of which follow a specific standard and structure. A sample data entry form source code is listed here.

frmIssueInv form

```
Option Explicit

'========  START of IMPLEMENTS ========================================='

' implementation interface for flat buttons
Implements IOwnerDrawButton
' implementation interface for listview
Implements IListViewOwner
' implementation interface for treeview
Implements ITreeViewOwner
Implements ICrystalEngine

'========  END of IMPLEMENTS ========================================='

'========   START of PRIVATE CONSTANTS   ============================='

Private Const ROOT_TEXT = "Issue Vouchers"                ' the text of the root node of treeview
Private Const FORM_CAPTION = "Issue Voucher(Push) "       ' initial form caption
Private Const INV_TYPE = 1

'========   END of PRIVATE CONSTANTS   ==============================='

'========   START of PRIVATE VARIABLES   ============================='

' array for storing treeview node id
Private m_NodePtr() As Long
' currently selected tree node
Private m_pSelNode As tagSelNodeOne
' user changed the data?
Private m_pUserChanged As Boolean
' is this the first time?
Private m_pFirstTime As Boolean
' count of records in tree
Private m_pRecCount As Long
Private m_PrintObject As clsPrintObject  ' print engine class pointer
Private ParamValInfo As PEValueInfo
' current focused entry field in form
Private m_pCurMainCtl As Control
' owner draw button class for drawing flat graphic buttons
Private m_cODBtn As cOwnerDrawButton
Private p_txtFields() As tagCtlProp
Private p_txtSearch() As tagCtlProp
Private p_dtFilter() As tagCtlProp

'========   END of PRIVATE VARIABLES   ==============================='

'========   START of PROPERTY VARIABLES   ============================='

Private m_pMainDirty As Boolean        ' main form is dirty?
Private m_pCurMode As Integer          ' currently in Edit or View mode?
Private m_pSubDirty As Boolean         ' subfor is dirty?
Private m_pCurrentCopy As Long

'========   END of PROPERTY VARIABLES   ==============================='

'========   START of PROPERTY PROCEDURES  ============================='

Public Property Get pMainDirty() As Boolean
```

```
   pMainDirty = m_pMainDirty
End Property

Public Property Let pMainDirty(parMainDirty As Boolean)
   m_pMainDirty = parMainDirty
End Property

Public Property Get pCurMode() As Integer
   pCurMode = m_pCurMode
End Property

Public Property Let pCurMode(parCurMode As Integer)
On Error GoTo Err_pCurMode

   m_pCurMode = parCurMode

   ' set the caption of the form depending on current edit state
   Select Case m_pCurMode
      Case MODE_ADD
         Caption = FORM_CAPTION & "+"

      Case MODE_VIEW
         Caption = FORM_CAPTION

      Case MODE_VIEW_AND_EDITING, MODE_ADD_AND_EDITING
         Caption = FORM_CAPTION & "*"

      Case MODE_READONLY
         Caption = FORM_CAPTION & MODE_READONLY_CAPTION

   End Select

   ' set state of buttons and controls according to form mode
   cboGroupBy.Enabled = (m_pCurMode = MODE_VIEW) Or (m_pCurMode = MODE_READONLY)
   cmdMain(0).Enabled = (m_pCurMode = MODE_VIEW)
   cmdMain(1).Enabled = (m_pCurMode = MODE_VIEW_AND_EDITING) Or _
                (m_pCurMode = MODE_ADD_AND_EDITING)
   cmdMain(2).Enabled = (m_pCurMode = MODE_VIEW_AND_EDITING) Or _
                (m_pCurMode = MODE_ADD_AND_EDITING)
   cmdMain(3).Enabled = (m_pCurMode = MODE_VIEW)
   cmdMain(4).Enabled = (m_pCurMode = MODE_VIEW) Or (m_pCurMode = MODE_READONLY)
   cmdMain(5).Enabled = (m_pCurMode = MODE_VIEW) Or (m_pCurMode = MODE_READONLY)
   cmdMain(6).Enabled = (m_pSelNode.TypeOfNode > 0) And (chkFields(0).Value = vbUnchecked)

Exit_pCurMode:
   Exit Property

Err_pCurMode:
   ProcessDBError
   Resume Exit_pCurMode
End Property

Public Property Get pSubDirty() As Boolean
   pSubDirty = m_pSubDirty
End Property

Public Property Let pSubDirty(parSubDirty As Boolean)
   m_pSubDirty = parSubDirty
End Property


'========  END of PROPERTY PROCEDURES  ============================='

'========  START of OBJECT EVENT PROCEDURES  ======================='

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' This procedure is called when the form is first loaded. This is the entry point.
'      1. Initialize listview, treeview, combo, and tab controls
'      2. Set form-level flags
```

```
'       3. Specify field names to which controls are bound
'       4. Fill the dropdown combos
'       5. Fill the tree with primary keys
'       4. If there are records, select the first one
'       5. Position the form
' Last Update Date: April 10, 2003
'
' Parameters: none
' Return Value: none
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''

Private Sub Form_Load()
On Error GoTo Err_Form_Load

Dim idx As Long
Dim RecCount As Long
Dim pWidth As String
Dim m_BasicTypeArray() As Variant
Dim m_GroupByArray() As Variant
Dim m_EmpArray() As Variant
Dim m_FacilityArray() As Variant
Dim m_IndDesigArray() As Variant
Dim m_ReasonArray() As Variant
Dim m_ItemsArray() As Variant
Dim ddwidth As Long

   ' it may take some time...
   Call Hourglass(hwnd, True)

   ' set form to pixel mode
   ScaleMode = vbPixels

   ' this form is implementing the treeview interface
   tvwMain.Attach Me
   ' attach imagelist to treeview
   tvwMain.SetImageList m_pImageBtn.hIml, TVSIL_NORMAL

   ' this form is implementing listview interface
   lvwSub.Attach Me
   ' attach imagelist to listview
   lvwSub.SetImageList m_pImageBtn.hIml, LVSIL_SMALL

   ' owner-drawn flat button interface object
   Set m_cODBtn = New cOwnerDrawButton
   ' this form is implementing flat button interface
   m_cODBtn.Attach Me
   ' buttons are in these containers
   m_cODBtn.AddhWnd picContainerSearch.hwnd
   m_cODBtn.AddhWnd picContainerFilter.hwnd
   m_cODBtn.AddhWnd Me.hwnd

   m_pFirstTime = True
   m_pUserChanged = True
   ' we are in view mode first
   pCurMode = MODE_VIEW
   ' the master and sub forms are not dirty initially
   m_pMainDirty = False
   m_pSubDirty = False

   ' insert 3 tabs in tab control - info, search and filter
   tabMain.AddTab BASIC_TAB, , , BASIC_TAB
   tabMain.AddTab SEARCH_TAB, , , SEARCH_TAB
   tabMain.AddTab FILTER_TAB, , , FILTER_TAB
   ' select first tab
   tabMain.SelectTab BASIC_TAB
   tabMain_TabClick 1

   ' only one item can be selected in listview
   lvwSub.SetStyle LVS_SINGLESEL
```

```
    ' enable grid lines and full row select in listview
    lvwSub.SetExtendedListViewStyle LVS_EX_GRIDLINES Or LVS_EX_FULLROWSELECT
    ' show the listview headers
    CreateLVWHeaders
    ' we want to catch the click event on header
    lvwSub.SubClassHeader

    ' initialize the array which will hold field information
    ReDim p_txtFields(txtFields.LBound To txtFields.UBound)
    ReDim p_txtSearch(txtSearch.LBound To txtSearch.UBound)
    ReDim p_dtFilter(dtFilter.LBound To dtFilter.UBound)

    ' set field data types for each control
    p_txtSearch(0).pDataFormat = dfinteger
    p_dtFilter(0).pDataFormat = dfSQLDate
    p_dtFilter(1).pDataFormat = dfSQLDate

    p_txtFields(0).pDataFormat = dfinteger
    p_txtFields(1).pDataFormat = dfSQLDate
    p_txtFields(2).pDataFormat = dfSQLDate
    p_txtFields(3).pDataFormat = dfSQLDate

    p_txtFields(1).pUserFormat = DATE_FORMAT
    p_txtFields(2).pUserFormat = DATE_FORMAT
    p_txtFields(3).pUserFormat = DATE_FORMAT
    p_dtFilter(0).pUserFormat = DATE_FORMAT
    p_dtFilter(1).pUserFormat = DATE_FORMAT
    txtSub(1).pUserFormat = NUMBER_FORMAT
    txtSub(3).pUserFormat = DATE_FORMAT
    txtSub(4).pUserFormat = DATE_FORMAT

    ' filter the data within a specific date range
    dtFilter(1) = ConvertDateDMY(Date)
    dtFilter(0) = ConvertDateDMY(DateAdd("d", -90, Date))

    ' set the field name that is bound to the control
    cboFields(0).pBoundField = "FacilityCode"
    cboFields(1).pBoundField = "PrepBy"
    cboFields(2).pBoundField = "AppBy1"
    cboFields(3).pBoundField = "IssuedBy"
    p_txtFields(0).pBoundField = "InvNo"
    p_txtFields(1).pBoundField = "PrepDate"
    p_txtFields(2).pBoundField = "AppDate1"
    p_txtFields(3).pBoundField = "IssuedDate"
    p_txtFields(4).pBoundField = "InvType"
    p_txtFields(5).pBoundField = "Remarks"

    ' store the associated field name in the tag
    chkFields(0).Tag = "bUpdated"

    txtSub(0).pBoundField = ""
    txtSub(1).pBoundField = "IssueQty"
    txtSub(2).pBoundField = "LotNo"
    txtSub(3).pBoundField = "MfgDate"
    txtSub(4).pBoundField = "ExpDate"
    txtSub(5).pBoundField = "CurStockQty"
    cboSub(0).pBoundField = "ItemCode"

    ' checkbox window has a property which handles dirty mode
    For idx = chkFields.LBound To chkFields.UBound
        SetProp chkFields(idx).hwnd, "IsDirty", False
    Next idx

    ' initialize the combo boxes
    pWidth = CLng(cboFields(0).Width / Screen.TwipsPerPixelX * 0.25) & ";" & CLng(cboFields(0).Width /
Screen.TwipsPerPixelX * 0.75)
    NewInitCombo cboFields(0), True, bbExtendedBound, 2, 0, 1, pWidth, dfText
    pWidth = "0;" & CLng(cboFields(1).Width / Screen.TwipsPerPixelX)
    NewInitCombo cboFields(1), True, bbExtendedBound, 2, 0, 1, pWidth, dfinteger
```

```
    pWidth = "0;" & CLng(cboFields(2).Width / Screen.TwipsPerPixelX)
    NewInitCombo cboFields(2), True, bbExtendedBound, 2, 0, 1, pWidth, dfinteger
    pWidth = "0;" & CLng(cboFields(3).Width)
    NewInitCombo cboFields(3), True, bbExtendedBound, 2, 0, 1, pWidth, dfinteger

    ddwidth = cboSub(0).DropDownWidth
    pWidth = CLng(ddwidth * 0.5) & ";0;" & CLng(ddwidth * 0.1) & ";" & CLng(ddwidth * 0.1) & ";0;" &
CLng(ddwidth * 0.1) & ";" & CLng(ddwidth * 0.2) & ";0"
    NewInitCombo cboSub(0), True, bbExtendedBound, 8, 2, 0, pWidth, dfText

    ' fill the combo boxes with values from database
    Call FillFormCombo(m_FacilityArray, m_EmpArray, m_IndDesigArray, m_ItemsArray)
    cboFields(0).SetUnboundData m_FacilityArray
    cboFields(1).SetUnboundData m_EmpArray
    cboFields(2).SetUnboundData m_EmpArray
    cboFields(3).SetUnboundData m_EmpArray
    cboSub(0).SetUnboundData m_ItemsArray

    ' data grouping options are...
    cboGroupBy.AddItem "Invoice No"
    cboGroupBy.AddItem "Facility"
    cboGroupBy.AddItem "Invoice Date"

    ' link the edit controls with the listview
    With lvwSub
        .EditEnabled = True
        .bEditing = False
        .ParenthWnd = Me.hwnd
        .AddControl cboSub(0), True, False
        .AddControl txtSub(1)
        .AddControl txtSub(0)
        .AddControl txtSub(5)
        .AddControl txtSub(2)
        .AddControl txtSub(3)
        .AddControl txtSub(4)
    End With

    ' fill the tree with primary keys
    cboGroupBy.ListIndex = 0

    If Not CanEditForm(Me.Name) Then
        LockFormEditing
    End If

    m_pCurrentCopy = 1

    Move 0, 0

Exit_Form_Load:
    m_pFirstTime = False

    Erase m_GroupByArray
    Erase m_EmpArray
    Erase m_FacilityArray
    Erase m_IndDesigArray
    Erase m_ItemsArray
    Call Hourglass(hwnd, False)
    Exit Sub

Err_Form_Load:
    ProcessDBError
    Resume Exit_Form_Load
End Sub

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' This procedure is called before the form is unloaded.
'       1. If the current record is not saved, try to save first
'       2. Get confirmation from user whether he or she really wants to Quit
'       3. If user does not want to quit, then set focus to the last control
```

```
'       4. If it is exit time, then release array and object memory
' Last Update Date: February 5, 2003
'
' Parameters:
'       Cancel—True will stop unloading of the form
'       UnloadMode—how unload was invoked
' Return Value: none
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''

Private Sub Form_QueryUnload(Cancel As Integer, UnloadMode As Integer)
On Error GoTo Err_Form_QueryUnload

Dim idx As Long

    ' the form is closing...
    ' if the for is dirty then try to save first and then quit
    If Not SaveMainRecord Then
        ' whether the user wants to quit without saving...
        If MsgBox("You have changes which will be lost if you Close now." & vbCrLf & _
                "Do you really want to close the form and discard changes?", vbQuestion + vbYesNo) = vbNo Then
            ' if not, return to current operations
            Cancel = True
            If Not m_pCurMainCtl Is Nothing Then
                m_pCurMainCtl.SetFocus
            End If
        End If
    End If

    ' save was successful or the form was not dirty...
    If Not Cancel Then
        Call Hourglass(hwnd, True)
        ' Hide the mess we're about to make...
        Visible = False
        Erase p_txtFields
        Erase p_txtSearch
        Erase p_dtFilter
        ' clear the treeview
        tvwMain.EmptyTreeView False
        Set m_cODBtn = Nothing
        ' clear the report object
        Set m_PrintObject = Nothing
        Set m_pCurMainCtl = Nothing
        ' delete tree nodes
        Erase m_NodePtr
        ' remove any property that we added to check box window
        For idx = chkFields.LBound To chkFields.UBound
            RemoveProp chkFields(idx).hwnd, "IsDirty"
        Next idx
        lvwSub.UnSubClassHeader
        Call Hourglass(hwnd, False)
    End If

Exit_Form_QueryUnload:
    Exit Sub

Err_Form_QueryUnload:
    ProcessDBError
    Resume Exit_Form_QueryUnload
End Sub

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' This procedure is called when the form gets resized. It positions
' the picture box containers that are holding controls
' Last Update Date: February 5, 2003
'
' Parameters: none
' Return Value: none
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
```

```
Private Sub Form_Resize()
On Error GoTo Err_Form_Resize

    ' resize the picture box containers
    With tabMain
        picContainerBasic.Move .ClientLeft, .ClientTop, .ClientWidth, .ClientHeight
        picContainerSearch.Move .ClientLeft, .ClientTop, .ClientWidth, .ClientHeight
        picContainerFilter.Move .ClientLeft, .ClientTop, .ClientWidth, .ClientHeight
    End With

Exit_Form_Resize:
    Exit Sub

Err_Form_Resize:
    ProcessDBError
    Resume Exit_Form_Resize
End Sub




''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' This procedure is called before user clicks on any of the tabs
' on the Tab control. It tries to save the editing before
' moving from one tab to another.
' Last Update Date: February 5, 2003
'
' Parameters: none
' Return Value: none
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''

Private Sub tabMain_BeforeClick(ByVal lTab As Long, bCancel As Boolean)
    ' try to save before changing tab
    bCancel = Not SaveSuccess
End Sub

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' This procedure is called when user clicks on any of the tabs
' on the Tab control. Depending on the tab, it makes one of the
' picture box container visible and sets focus to a specific control.
'
' Last Update Date: February 5, 2003
'
' Parameters:
'   lTab—index of tab that was clicked
' Return Value: none
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''

Private Sub tabMain_TabClick(ByVal lTab As Long)
On Error GoTo Err_tabMain_Click

    ' make the appropriate picture container visible
    picContainerBasic.Visible = (lTab = 1)
    picContainerSearch.Visible = (lTab = 2)
    picContainerFilter.Visible = (lTab = 3)
    ' set focus to specific control
    If lTab = 2 Then
        txtSearch(0).SetFocus
    ElseIf lTab = 3 Then
        dtFilter(0).SetFocus
    End If

Exit_tabMain_Click:
    Exit Sub

Err_tabMain_Click:
    ProcessDBError
    Resume Exit_tabMain_Click
```

```
End Sub

'''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' Click event handler for command buttons
' Last Update Date: 05 Feb, 2003
'
' Parameters:
'       Index—index of the button in the button array
' Return Value: none
'''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''

Private Sub cmdMain_Click(Index As Integer)
On Error GoTo Err_cmdMain_Click

    Select Case Index
        Case 0                  ' new record
            AddNewMainRecord

        Case 1                  ' save record
            SaveSuccess

        Case 2                  ' undo record
            UndoMainRecord

        Case 3                  ' delete record
            DeleteMainRecord

        Case 4                  ' preview report
            picPrintOptions.Visible = True

        Case 5                  ' close form
            UnloadMainForm

        Case 6                  ' post to stock
            PostToStock

    End Select

Exit_cmdMain_Click:
    Exit Sub

Err_cmdMain_Click:
    ProcessDBError
    Resume Exit_cmdMain_Click
End Sub

'''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' Click event handler for Search button.
'       1. Gets the searched invoice from database
'       2. Try to find the invoice in the tree
'       3. If found, select the tree node
' Last Update Date: February 5, 2003
'
' Parameters:
'       Index—index of the button in the button array
' Return Value: none
'''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''

Private Sub cmdSearch_Click()
On Error GoTo Err_cmdSearch_Click

Dim RecCount As Long
Dim nodex As Long
Dim recArray() As Variant

    ' search for invoice no.
    If txtSearch(0) <> "" Then
        ' try to find the invoice no. from database
        Call GetTreeWoList(recArray, RecCount, 9999, CStr(dtFilter(0)), CStr(dtFilter(1)), CLng(txtSearch(0)))
```

```
        ' if a matching record is found...
        If RecCount > 0 Then
            ' find the node and select it in the tree
            nodex = FindNodeInTree(CLng(recArray(0, 0)))
            If nodex <> 0 Then
                tvwMain.SelectItem nodex
            End If
        Else
            MsgBox "Invoice Not Found", vbInformation
        End If
    End If

Exit_cmdSearch_Click:
    Erase recArray
    Exit Sub

Err_cmdSearch_Click:
    ProcessDBError
    Resume Exit_cmdSearch_Click
End Sub

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' Click event handler for Filter button; refills the tree with
' records from database based on filter criteria.
' Last Update Date: February 5, 2003
'
' Parameters: none
' Return Value: none
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''

Private Sub cmdFilter_Click()
    RefreshTree True
End Sub

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' This procedure runs when user presses any key in a text box.
' Special handling is done for ENTER and ESC keys
' Last Update Date: February 5, 2003
'
' Parameters:
'       Index—index of text box in control array
'       KeyAscii—integer value of keyboard key pressed
' Return Value: none
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''

Private Sub txtFields_KeyPress(Index As Integer, KeyAscii As Integer)
    Main_KeyPress Index, KeyAscii, txtFields(Index).MultiLine
End Sub

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' This procedure runs when user presses any key in a check box
' Special handling is done for ENTER and ESC keys
' Last Update Date: February 5, 2003
'
' Parameters:
'       Index—index of check box in control array
'       KeyAscii—integer value of keyboard key pressed
' Return Value: none
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''

Private Sub chkFields_KeyPress(Index As Integer, KeyAscii As Integer)
    Main_KeyPress Index, KeyAscii
End Sub

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' This procedure runs when user presses any key in a combo box
' Special handling is done for ENTER and ESC keys
' Last Update Date: February 5, 2003
```

```
'
' Parameters:
'       Index—index of text box in control array
'       KeyAscii—integer value of keyboard key pressed
' Return Value: none
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''

Private Sub cboFields_KeyPress(Index As Integer, KeyAscii As Integer)
    Main_KeyPress Index, KeyAscii
End Sub

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' This procedure is called when any editing is done in a text box
' If user is editing in the text box, then set its Dirty flag
' Also set the form flags
' Last Update Date: February 5, 2003
'
' Parameters:
'       Index—index of text box in control array
' Return Value: none
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''

Private Sub txtFields_Change(Index As Integer)
    If m_pUserChanged Then
        p_txtFields(Index).IsDirty = True
    End If
    Main_Change
End Sub

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' This procedure is called when any editing is done in a combo box
' If user is editing in the combo box, then set its Dirty flag
' Also set the form flags
' Last Update Date: February 5, 2003
'
' Parameters:
'       Index—index of text box in control array
' Return Value: none
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''

Private Sub cboFields_Change(Index As Integer)
    Main_Change
End Sub

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' This procedure is called when user clicks on a check box
' If user has clicked in the combo box, then set its Dirty flag
' Also set the form flags
' Last Update Date: February 5, 2003
'
' Parameters:
'       Index—index of check box in control array
' Return Value: none
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''

Private Sub chkFields_Click(Index As Integer)
    Main_Change
    If m_pUserChanged Then
        ' check box is dirty, so set flag
        SetProp chkFields(Index).hwnd, "IsDirty", True
    End If
    If (Index = 0) Then
        lvwSub.EditEnabled = (chkFields(Index).Value = vbUnchecked)
        txtFields(0).Enabled = (chkFields(Index).Value = vbUnchecked)
        txtFields(1).Enabled = (chkFields(Index).Value = vbUnchecked)
        txtFields(2).Enabled = (chkFields(Index).Value = vbUnchecked)
        txtFields(3).Enabled = (chkFields(Index).Value = vbUnchecked)
        cboFields(0).Enabled = (chkFields(Index).Value = vbUnchecked)
        cboFields(1).Enabled = (chkFields(Index).Value = vbUnchecked)
```

```
      cboFields(2).Enabled = (chkFields(Index).Value = vbUnchecked)
      cboFields(3).Enabled = (chkFields(Index).Value = vbUnchecked)
   End If
End Sub

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' This procedure is called when user selects an item in a
' combo box
If user has selected an item, then set its Dirty flag
' Also set the form flags
' Last Update Date: February 5, 2003
'
' Parameters:
'      Index - index of combo box in control array
' Return Value: none
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''

Private Sub cboFields_Click(Index As Integer)
   Main_Change
End Sub

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' This procedure is called when focus is moving from text box
' If the text box has formatting set and it has values,
' then the formatting is applied on the data
' Last Update Date: February 5, 2003
'
' Parameters:
'      Index—index of combo box in control array
' Return Value: none
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''

Private Sub txtFields_LostFocus(Index As Integer)
Dim m_text As String

   If p_txtFields(Index).IsDirty Then
      If p_txtFields(Index).pDataFormat = dfSQLDate Then
         m_text = txtFields(Index)
         If m_text <> "" Then
            txtFields(Index) = Format(m_text, p_txtFields(Index).pUserFormat)
         End If
      End If
   End If

End Sub

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' This procedure is called before focus is leaving from a text box.
'      1. If the text format is numeric, then check whether it's a valid number
'      2. If the text format is date type, then check whether it's
'         a valid date
'      3. If the data are not valid, then set Calcel flag to true to prevent
'         the focus from leaving the text box
' Last Update Date: February 5, 2003
'
' Parameters:
'      Index—index of text box in control array
'      Cancel —rue will stop the focus from leaving the control
' Return Value: none
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''

Private Sub txtFields_Validate(Index As Integer, Cancel As Boolean)
On Error GoTo Err_txtFields_Validate

   Set m_pCurMainCtl = txtFields(Index)
   With txtFields(Index)
   If VBA.Trim$(.text) <> "" Then
      If (p_txtFields(Index).pDataFormat = dfinteger Or p_txtFields(Index).pDataFormat = dfDouble) And _
            (Not IsNumeric(.text)) Then
```

```
            MsgBox "Pls Enter a Numeric Value", vbInformation
            Cancel = True
            m_pCurMainCtl.SetFocus
        ElseIf (p_txtFields(Index).pDataFormat = dfSQLDate) And (Not IsValidDate(.text)) Then
            MsgBox "Pls Enter a Date", vbInformation
            Cancel = True
            m_pCurMainCtl.SetFocus
        End If
    End If
    End With

Exit_txtFields_Validate:
    Exit Sub

Err_txtFields_Validate:
    ProcessDBError
    Resume Exit_txtFields_Validate
End Sub

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' This procedure is called before focus is leaving from a combo box
' It just keeps the reference of the combo box to a variable for later use
' Last Update Date: February 5, 2003
'
' Parameters:
'       Index—index of combo box in control array
'       Cancel—True will stop the focus from leaving the control
' Return Value: none
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''

Private Sub cboFields_Validate(Index As Integer, Cancel As Boolean)
    ' set the current focus control
'    Set m_pCurMainCtl = cboFields(Index)
End Sub

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' This procedure runs when user presses any key in the Search text box
' Special handling is done for ENTER and ESC keys
' Last Update Date: February 5, 2003
'
' Parameters:
'       Index—index of text box in control array
'       KeyAscii—integer value of keyboard key pressed
' Return Value: none
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''

Private Sub txtSearch_KeyPress(Index As Integer, KeyAscii As Integer)
    If (KeyAscii = 10 Or KeyAscii = 13) Then
        SendKeys "{TAB}"
        KeyAscii = 0
    End If
End Sub

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' This procedure runs when user presses any key in filter date text boxes
' Special handling is done for ENTER and ESC keys
' Last Update Date: February 5, 2003
'
' Parameters:
'       Index—index of text box in control array
'       KeyAscii—integer value of keyboard key pressed
' Return Value: none
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''

Private Sub dtFilter_KeyPress(Index As Integer, KeyAscii As Integer)
    If (KeyAscii = 10 Or KeyAscii = 13) Then
        SendKeys "{TAB}"
        KeyAscii = 0
    End If
```

```
End Sub

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' This procedure is called before focus is leaving from Search text box.
'       1. If the text format is numeric, then check whether it's a valid number
'       2. If the text format is date type, then check whether it's
'          a valid date
'       3. If the data are not valid, then set Calcel flag to true to prevent
'          the focus from leaving the text box
' Last Update Date: February 5, 2003
'
' Parameters:
'       Index—index of text box in control array
'       Cancel—True will stop the focus from leaving the control
' Return Value: none
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''

Private Sub txtSearch_Validate(Index As Integer, Cancel As Boolean)
On Error GoTo Err_txtSearch_Validate

    With txtSearch(Index)
    If VBA.Trim$(.text) <> "" Then
        If (p_txtSearch(Index).pDataFormat = dfinteger Or p_txtSearch(Index).pDataFormat = dfDouble) And _
                (Not IsNumeric(.text)) Then
            MsgBox "Enter a Numeric Value", vbInformation
            Cancel = True
            .SetFocus
        End If
    End If
    End With

Exit_txtSearch_Validate:
    Exit Sub

Err_txtSearch_Validate:
    ProcessDBError
    Resume Exit_txtSearch_Validate
End Sub

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' This procedure is called before focus is leaving from date filter text box.
'       1. If the text format is numeric, then check whether it's a valid number
'       2. If the text format is date type, then check whether it's
'          a valid date
'       3. If the data are not valid, then set Calcel flag to true to prevent
'          the focus from leaving the text box
' Last Update Date: February 5, 2003
'
' Parameters:
'       Index—index of text box in control array
'       Cancel—True will stop the focus from leaving the control
' Return Value: none
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''

Private Sub dtFilter_Validate(Index As Integer, Cancel As Boolean)
On Error GoTo Err_dtFilter_Validate

    With dtFilter(Index)
    If VBA.Trim$(.text) <> "" Then
        If (p_dtFilter(Index).pDataFormat = dfSQLDate) And (Not IsValidDate(.text)) Then
            MsgBox "Pls Enter a Date", vbInformation
            Cancel = True
            .SetFocus
        End If
    End If
    End With

Exit_dtFilter_Validate:
    Exit Sub
```

```
Err_dtFilter_Validate:
   ProcessDBError
   Resume Exit_dtFilter_Validate
End Sub


Private Sub cboGroupBy_Click()
On Error GoTo Err_cboGroupBy_Click

   If SaveSuccess Then
       RefreshTree True
   End If

Exit_cboGroupBy_Click:
   Exit Sub

Err_cboGroupBy_Click:
   ProcessDBError
   Resume Exit_cboGroupBy_Click
End Sub

Private Sub cboGroupBy_GotFocus()
   SaveSuccess
End Sub

Private Sub txtSub_Change(Index As Integer)
   Sub_Change
End Sub

Private Sub cboSub_Click(Index As Integer)
Dim bLock As Boolean

   Sub_Change
   If Index = 0 Then
      If cboSub(0) <> "" Then
         txtSub(0) = cboSub(0).Column(7)
         txtSub(5) = cboSub(0).Column(6)
         txtSub(5).IsDirty = True
         txtSub(2) = cboSub(0).Column(3)
         txtSub(2).IsDirty = True
         txtSub(3) = cboSub(0).Column(4)
         txtSub(3).IsDirty = True
         txtSub(4) = cboSub(0).Column(5)
         txtSub(4).IsDirty = True
      End If
   End If

End Sub

Private Sub txtSub_KeyPress(Index As Integer, KeyAscii As Integer, Shift As Integer)
   Sub_KeyPress Index, KeyAscii
End Sub

Private Sub cboSub_KeyPress(Index As Integer, KeyAscii As Integer)
   Sub_KeyPress Index, KeyAscii
End Sub

Private Sub txtSub_LostFocus(Index As Integer)
   lvwSub.ProcessLostFocus Screen.ActiveControl.hwnd, txtSub(Index).hwnd, tvwMain.hwnd
End Sub

Private Sub cboSub_LostFocus(Index As Integer)
   lvwSub.ProcessLostFocus Screen.ActiveControl.hwnd, cboSub(Index).hwnd, tvwMain.hwnd
End Sub

Private Sub txtSub_Validate(Index As Integer, Cancel As Boolean)
On Error GoTo Err_txtSub_Validate
```

```
    With txtSub(Index)
    ' if the field contains text
    If VBA.Trim$(.text) <> "" Then
        ' if format is numeric
        If (.pDataFormat = dfinteger Or .pDataFormat = dfDouble) And _
              (Not IsNumeric(.text)) Then
            MsgBox "Enter a Numeric Value", vbInformation
            Cancel = True
            .SetFocus
        ' if format is date
        ElseIf (.pDataFormat = dfSQLDate) And (Not IsValidDate(.text)) Then
            MsgBox "Pls Enter a Date", vbInformation
            Cancel = True
            .SetFocus
        End If
    End If
    End With

Exit_txtSub_Validate:
    Exit Sub

Err_txtSub_Validate:
    ProcessDBError
    Resume Exit_txtSub_Validate
End Sub

Private Sub cmdSub_Click(Index As Integer)
    Select Case Index
        Case 0      ' add
            If (m_pSelNode.TypeOfNode > 0) And (chkFields(0).Value = vbUnchecked) Then
                AddNewSubRecord
            End If
        Case 1      ' delete
            If (m_pSelNode.TypeOfNode > 0) And (chkFields(0).Value = vbUnchecked) Then
                DeleteSubRecord
            End If
    End Select
End Sub

'========   END of OBJECT EVENT PROCEDURES  ========================='

'========   START of USER DEFINED FUNCTIONS ========================='

Private Sub RefreshTree(brefresh As Boolean)
On Error GoTo Err_RefreshTree

Dim recArray() As Variant

    Select Case cboGroupBy.ListIndex
        Case 0
            Call GetTreeWoList(recArray, m_pRecCount, 0, CStr(dtFilter(0)), CStr(dtFilter(1)))
            PopulateTreeMain tvwMain, recArray, m_pRecCount, m_NodePtr, brefresh

        Case 1
            Call GetTreeWoList(recArray, m_pRecCount, 1, CStr(dtFilter(0)), CStr(dtFilter(1)))
            PopulateTreeMain tvwMain, recArray, m_pRecCount, m_NodePtr, brefresh, True

        Case 2
            Call GetTreeWoList(recArray, m_pRecCount, 2, CStr(dtFilter(0)), CStr(dtFilter(1)))
            PopulateTreeMain tvwMain, recArray, m_pRecCount, m_NodePtr, brefresh, True

    End Select

    UpdateTotalNode
    SetFocusAPI tvwMain.GetContainerHWnd

Exit_RefreshTree:
    Erase recArray
    Exit Sub
```

```
Err_RefreshTree:
   ProcessDBError
   Resume Exit_RefreshTree
End Sub

Private Sub ClearControls()
On Error GoTo Err_ClearControls

Dim idx As Integer

   m_pUserChanged = False

   For idx = txtFields.LBound To txtFields.UBound
      txtFields(idx) = ""
   Next idx

   For idx = chkFields.LBound To chkFields.UBound
      SetCheckValue chkFields(idx).hwnd, Null
   Next idx

   For idx = cboFields.LBound To cboFields.UBound
      cboFields(idx) = Null
   Next idx

   lvwSub.DeleteAllItems

   m_pUserChanged = True

Exit_ClearControls:
   Exit Sub

Err_ClearControls:
   ProcessDBError
   Resume Exit_ClearControls
End Sub


Private Sub FillOneInfo(recArray As Variant, RecCount As Long)
On Error GoTo Err_FillOneInfo

Dim idx As Integer

   If m_pSelNode.SelNodeId = -1 Then
      Exit Sub
   End If

   If RecCount = 1 Then
      m_pUserChanged = False

      txtFields(0) = recArray(0, 0)
      cboFields(0) = recArray(1, 0)
      cboFields(1) = recArray(2, 0)
      txtFields(1) = recArray(3, 0)
      cboFields(2) = recArray(4, 0)
      txtFields(2) = nz(recArray(5, 0), "")
      cboFields(3) = recArray(6, 0)
      txtFields(3) = nz(recArray(7, 0), "")
      txtFields(5) = nz(recArray(9, 0), "")

      SetCheckValue chkFields(0).hwnd, recArray(8, 0)

      m_pUserChanged = True

   End If

Exit_FillOneInfo:
   Exit Sub
```

```
Err_FillOneInfo:
   ProcessDBError
   Resume Exit_FillOneInfo
End Sub

Private Sub FillManyInfo(recArray As Variant, RecCount As Long)
On Error GoTo Err_FillManyInfo

Dim idx As Long
Dim totAmt As Double

   lvwSub.DeleteAllItems

   For idx = 0 To RecCount - 1
      lvwSub.AddItem recArray(2, idx), 12
      lvwSub.SetTag idx, 0, CLng(SaveStringToMemory(CStr(recArray(1, idx))))

      lvwSub.AddSubItem idx, 1, Format$(nz(recArray(5, idx), 0), NUMBER_FORMAT)
      lvwSub.SetTag idx, 1, CLng(recArray(0, idx))

      lvwSub.AddSubItem idx, 2, CStr(recArray(3, idx))
      lvwSub.AddSubItem idx, 3, Format$(nz(recArray(9, idx), 0), NUMBER_FORMAT)
      lvwSub.AddSubItem idx, 4, CStr(nz(recArray(7, idx), ""))
      lvwSub.AddSubItem idx, 5, Format$(nz(recArray(8, idx), ""), DATE_FORMAT)
      lvwSub.AddSubItem idx, 6, Format$(nz(recArray(6, idx), ""), DATE_FORMAT)
   Next idx

   CalcSubTotFields

Exit_FillManyInfo:
   Exit Sub

Err_FillManyInfo:
   MsgBox Err.description
   Resume Exit_FillManyInfo
End Sub

Private Sub CreateLVWHeaders()
On Error GoTo Err_CreateLVWHeaders

   lvwSub.SetHeaderFont 18
   lvwSub.AddColumn 0, "Item", 155, LVCFMT_LEFT
   lvwSub.AddColumn 1, "Issued Qty", 75, LVCFMT_RIGHT
   lvwSub.AddColumn 2, "Unit", 35, LVCFMT_LEFT
   lvwSub.AddColumn 3, "Stock Qty", 75, LVCFMT_RIGHT
   lvwSub.AddColumn 4, "Lot No", 100, LVCFMT_LEFT
   lvwSub.AddColumn 5, "Mfg. Date", 80, LVCFMT_LEFT
   lvwSub.AddColumn 6, "Expiry Date", 80, LVCFMT_LEFT

Exit_CreateLVWHeaders:
   Exit Sub

Err_CreateLVWHeaders:
   MsgBox Err.description
   Resume Exit_CreateLVWHeaders
End Sub


Private Sub AddNewMainRecord()
On Error GoTo Err_AddNewMainRecord

   If pCurMode <> MODE_ADD And SaveSuccess Then
      ' we are entering add mode
      pCurMode = MODE_ADD
      ' clear the controls
      ClearControls
      ' initialize the fields
      InitOneFields
      ' recalc the sum fields
```

```
      CalcSubTotFields
      ' select the frst tab
      If tabMain.SelectedTab <> 1 Then
         tabMain.SelectTab BASIC_TAB
         tabMain_TabClick 1
      End If
      cboFields(0).SetFocus
   End If

Exit_AddNewMainRecord:
   Exit Sub

Err_AddNewMainRecord:
   ProcessDBError
   Resume Exit_AddNewMainRecord

End Sub

Private Sub UndoMainRecord()
On Error GoTo Err_UndoMainRecord

   Select Case pCurMode
      Case MODE_VIEW_AND_EDITING, MODE_ADD, MODE_ADD_AND_EDITING
         If tvwMain.GetCount > 0 Then
            'reset to view mode
            pCurMode = MODE_VIEW
            pMainDirty = False
            'mimic node selection
            ITreeViewOwner_AfterSelectionChange m_pSelNode.SelTreeNodePtr
         End If
   End Select

Exit_UndoMainRecord:
   Exit Sub

Err_UndoMainRecord:
   ProcessDBError
   Resume Exit_UndoMainRecord
End Sub


Private Sub DeleteMainRecord()
On Error GoTo Err_DeleteMainRecord

Dim strsql As String
Dim RecCount As Long
Dim nodex As Long
Dim nextselptr As Long
Dim cData As clsGetData

   Select Case pCurMode
      Case MODE_VIEW, MODE_VIEW_AND_EDITING, MODE_ADD_AND_EDITING
         If m_pSelNode.SelNodeId = -1 Then
            Exit Sub
         End If

         If SaveSuccess Then
            If MsgBox("Do you really want to delete the selected Invoice?", vbYesNo + vbQuestion +
vbDefaultButton2) = vbYes Then
               strsql = "delete from Indent " & _
                     " where InvType=" & INV_TYPE & " and InvNo=" & m_pSelNode.SelNodeId
               Set cData = New clsGetData
               Call cData.RunQuery(m_ConStr, strsql, RecCount, CURRENT_USERID, CURRENT_COMPUTERNAME)

               If RecCount > 0 Then

                  m_pRecCount = m_pRecCount - 1
                  UpdateTotalNode
```

```
                ' change the node selection in tree
                nodex = m_pSelNode.SelTreeNodePtr
                If tvwMain.GetNextSibling(nodex) = 0 Then
                   ' if no sibling (brother) next then look for one before
                   If tvwMain.GetPrevSibling(nodex) = 0 Then
                      ' select the parent of the deleted node
                      nextselptr = tvwMain.GetParentNode(nodex)
                   Else
                      ' if sibling before, then select
                      nextselptr = tvwMain.GetPrevSibling(nodex)
                   End If
                Else
                   ' if there is a sibling (brother) next, then select
                   nextselptr = tvwMain.GetNextSibling(nodex)
                End If
                ' delete the node from tree
                tvwMain.DeleteItem nodex
                ' select the node in the tree
                tvwMain.SelectItem nextselptr
             End If
          End If
       End If
   End Select

Exit_DeleteMainRecord:
   Set cData = Nothing
   Exit Sub

Err_DeleteMainRecord:
   ProcessDBError
   Resume Exit_DeleteMainRecord
End Sub


Private Sub PreviewRecord(pCopyNumber As Integer)
On Error GoTo Err_PreviewRecord

   If Not m_PrintObject Is Nothing Then
      Set m_PrintObject = Nothing
   End If
   Set m_PrintObject = New clsPrintObject
   With m_PrintObject
      .Attach Me
      ' print engine handle
      .EngineHandle = m_prnHandle
      .UID = REPORT_UID
      .PWD = REPORT_PWD
      ' parent window of print preview window
      .ParenthWnd = frmMain.hwnd
      .ReportName = App.Path & "\Reports\IssueInv.rpt"
      .WindowTitle = "Report - Indent Voucher"
      ' show the report preview

      If pCopyNumber = 0 Then
         m_pCurrentCopy = 1
         Do
            .PrintReportDirectly
            m_pCurrentCopy = m_pCurrentCopy + 1
         Loop While m_pCurrentCopy <= 5
      Else
         m_pCurrentCopy = pCopyNumber
         .PrintReport
      End If
   End With

Exit_PreviewRecord:
   Exit Sub

Err_PreviewRecord:
```

```
    ProcessDBError
    Resume Exit_PreviewRecord
End Sub

Private Sub UnloadMainForm()
    Unload Me
End Sub

' Check the saving status; if not successful, then get focus back
' to the previous control
Private Function SaveSuccess() As Boolean
On Error GoTo Err_SaveSuccess

    SaveSuccess = True

    SaveSuccess = SaveMainRecord
    If (Not SaveSuccess) And (Not m_pCurMainCtl Is Nothing) Then
        m_pCurMainCtl.SetFocus
    End If

Exit_SaveSuccess:
    Exit Function

Err_SaveSuccess:
    ProcessDBError
    Resume Exit_SaveSuccess
End Function

Private Function SaveMainRecord() As Boolean
On Error GoTo Err_SaveMainRecord

Dim idx As Integer
Dim nodex As Long
Dim lastText As Long
Dim lastType As String

    SaveMainRecord = False

    ' if the form is not dirty, then nothing to do
    If Not pMainDirty Then
        SaveMainRecord = True
        Exit Function
    End If

    Select Case pCurMode
        Case MODE_VIEW_AND_EDITING
            If SaveRecord Then
                ' if user changed the field that is shown as node label, then update the label
                If txtFields(0) <> tvwMain.GetNodeText(m_pSelNode.SelTreeNodePtr) Then
                    UpdateNode m_pSelNode.SelTreeNodePtr, CStr(txtFields(0))
                End If
                pCurMode = MODE_VIEW
                SaveMainRecord = True
                pMainDirty = False
            End If

        Case MODE_ADD_AND_EDITING
            If SaveRecord Then
                lastText = txtFields(0)
                pCurMode = MODE_VIEW
                pMainDirty = False
                RefreshTree False

                ' find the newly added node and select
                nodex = FindNodeInTree(lastText)
                If nodex <> 0 Then
                    tvwMain.SelectItem nodex
                End If
                SaveMainRecord = True
```

```
        End If
    End Select

Exit_SaveMainRecord:
    Exit Function

Err_SaveMainRecord:
    ProcessDBError
    Resume Exit_SaveMainRecord
End Function

Private Function SaveRecord() As Boolean
On Error GoTo Err_SaveRecord

Dim strsql As String
Dim RecCount As Long
Dim updstring As String
Dim insstring As String
Dim cData As clsGetData

    SaveRecord = False

    If Not RecordIsValid Then
        Exit Function
    End If

    Select Case pCurMode
        Case MODE_ADD_AND_EDITING
            ' try to insert the new record
            insstring = CreateOneInsertString
            strsql = "insert into Indent " & insstring

            Set cData = New clsGetData
            Call cData.RunQuery(m_ConStr, strsql, RecCount, CURRENT_USERID, CURRENT_COMPUTERNAME)

            If RecCount > 0 Then
                ' if save was successful then reset the dirty flags of entry fields
                ResetDirtyFlag
                SaveRecord = True
            End If

        Case MODE_VIEW_AND_EDITING
            updstring = CreateOneUpdateString
            If m_pSelNode.SelNodeId <> -1 Then
                If updstring <> "" Then
                    ' if need to update, then update
                    strsql = "update Indent set " & updstring & _
                            " where InvType=" & INV_TYPE & " and InvNo=" & m_pSelNode.SelNodeId

                    Set cData = New clsGetData
                    Call cData.RunQuery(m_ConStr, strsql, RecCount, CURRENT_USERID, CURRENT_COMPUTERNAME)

                    If RecCount > 0 Then
                        ResetDirtyFlag
                        SaveRecord = True
                    End If
                Else
                    ResetDirtyFlag
                    SaveRecord = True
                End If
            End If
    End Select

Exit_SaveRecord:
    Set cData = Nothing
    Exit Function

Err_SaveRecord:
    ProcessDBError
```

```
    Resume Next
End Function

' Create a string that will be used for updating the record
Private Function CreateOneUpdateString() As String
On Error GoTo Err_CreateOneUpdateString

Dim idx As Integer
Dim strsql As String

    strsql = ""
    For idx = txtFields.LBound To txtFields.UBound
        ' if the control is dirty then include it in the update string
        If p_txtFields(idx).IsDirty Then
            strsql = strsql & p_txtFields(idx).pBoundField & "=" & _
                mGetDBText(txtFields(idx), p_txtFields(idx).pDataFormat, p_txtFields(idx).pUserFormat) & ","
        End If
    Next idx

    For idx = cboFields.LBound To cboFields.UBound
        ' if the control is dirty then include it in the update string
        If cboFields(idx).IsDirty Then
            strsql = strsql & cboFields(idx).pBoundField & "=" & cboFields(idx).mGetDBText & ","
        End If
    Next idx

    ' if check box is dirty
    For idx = chkFields.LBound To chkFields.UBound
        If GetProp(chkFields(idx).hwnd, "IsDirty") Then
            strsql = strsql & chkFields(idx).Tag & "=" & GetCheckValue(chkFields(idx).hwnd) & ","
        End If
    Next idx

    If Right$(strsql, 1) = "," Then
        strsql = Left$(strsql, Len(strsql) - 1)
    End If

    CreateOneUpdateString = strsql

Exit_CreateOneUpdateString:
    Exit Function

Err_CreateOneUpdateString:
    ProcessDBError
    Resume Exit_CreateOneUpdateString

End Function

' Create the string that will be used for inserting records into the table
Private Function CreateOneInsertString() As String
On Error GoTo Err_CreateOneInsertString

Dim idx As Integer
Dim strsql As String
Dim strfields As String
Dim strvals As String

    For idx = txtFields.LBound To txtFields.UBound
        ' if the control is dirty, then include in the insert string
        If p_txtFields(idx).pBoundField <> "" Then
            strfields = strfields & p_txtFields(idx).pBoundField & ","
            strvals = strvals & mGetDBText(txtFields(idx), p_txtFields(idx).pDataFormat,
p_txtFields(idx).pUserFormat) & ","
        End If
    Next idx

    For idx = cboFields.LBound To cboFields.UBound
        ' if the control is dirty, then include in the insert string
        If cboFields(idx).pBoundField <> "" Then
```

```
            strfields = strfields & cboFields(idx).pBoundField & ","
            strvals = strvals & cboFields(idx).mGetDBText & ","
        End If
    Next idx

    For idx = chkFields.LBound To chkFields.UBound
        strfields = strfields & chkFields(idx).Tag & ","
        strvals = strvals & GetCheckValue(chkFields(idx).hwnd) & ","
    Next idx

    If Right$(strfields, 1) = "," Then
        strfields = Left$(strfields, Len(strfields) - 1)
    End If

    If Right$(strvals, 1) = "," Then
        strvals = Left$(strvals, Len(strvals) - 1)
    End If

    If strfields <> "" And strvals <> "" Then
        CreateOneInsertString = "(" & strfields & ") values (" & strvals & ")"
    End If

Exit_CreateOneInsertString:
    Exit Function

Err_CreateOneInsertString:
    ProcessDBError
    Resume Exit_CreateOneInsertString

End Function

' Reset all the dirty controls to normal state
Private Sub ResetDirtyFlag()
On Error GoTo Err_ResetDirtyFlag

Dim idx As Integer

    For idx = txtFields.LBound To txtFields.UBound
        If p_txtFields(idx).IsDirty Then
            p_txtFields(idx).IsDirty = False
        End If
    Next idx

    For idx = cboFields.LBound To cboFields.UBound
        If cboFields(idx).IsDirty Then
            cboFields(idx).IsDirty = False
        End If
    Next idx

    For idx = chkFields.LBound To chkFields.UBound
        SetProp chkFields(idx).hwnd, "IsDirty", False
    Next idx

Exit_ResetDirtyFlag:
    Exit Sub

Err_ResetDirtyFlag:
    ProcessDBError
    Resume Exit_ResetDirtyFlag

End Sub


Private Function AddBlankItem() As Long
On Error GoTo Err_AddBlankItem

Dim idx As Integer
Dim nextid As Long
```

```
   AddBlankItem = 0

   nextid = lvwSub.Count
   lvwSub.AddItem "", 12

   For idx = 1 To 6
      lvwSub.AddSubItem nextid, idx, ""
   Next idx

   AddBlankItem = nextid

Exit_AddBlankItem:
   Exit Function

Err_AddBlankItem:
   ProcessDBError
   Resume Exit_AddBlankItem
End Function


Private Function SaveSubRecord(LastIndex As Long) As Boolean
On Error GoTo Err_SaveSubRecord

Dim strsql As String
Dim RecCount As Long
Dim updstring As String
Dim insstring As String
Dim cData As clsGetData
Dim newID As Long

   SaveSubRecord = False

   ' if the form is not dirty, then nothing to do
   If Not pSubDirty Then
      SaveSubRecord = True
      Exit Function
   End If

   If Not SubRecordIsValid Then
      Exit Function
   End If

   If Not CanSaveSubRecord Then
      Exit Function
   End If

   Select Case pCurMode
      Case MODE_ADD_AND_EDITING
         ' try to insert the new record
         insstring = CreateSubInsertString(LastIndex)
         strsql = "insert into IndentItems " & insstring
         Set cData = New clsGetData
         Call cData.InsertRecIdentity(m_ConStr, strsql, newID, "ItemSl", RecCount, CURRENT_USERID,
CURRENT_COMPUTERNAME)

         If RecCount > 0 Then
            lvwSub.SetTag LastIndex, 1, newID
            SaveSubRecord = True
            pSubDirty = False
            pCurMode = MODE_VIEW
         End If

      Case MODE_VIEW_AND_EDITING
         updstring = CreateSubUpdateString
         If updstring <> "" Then
            ' if need to update, then update
            strsql = "update IndentItems set " & updstring & _
                  " where ItemSl=" & lvwSub.GetTag(LastIndex, 1)
```

```
            Set cData = New clsGetData
            Call cData.RunQuery(m_ConStr, strsql, RecCount, CURRENT_USERID, CURRENT_COMPUTERNAME)
            If RecCount > 0 Then
               SaveSubRecord = True
               pSubDirty = False
               pCurMode = MODE_VIEW
            End If
         Else
            SaveSubRecord = True
            pSubDirty = False
            pCurMode = MODE_VIEW
         End If
   End Select

Exit_SaveSubRecord:
   Set cData = Nothing
   Exit Function

Err_SaveSubRecord:
   ProcessDBError
   Resume Next
End Function

' Create a string whcih will be used for updating the record
Private Function CreateSubUpdateString() As String
On Error GoTo Err_CreateSubUpdateString

Dim idx As Integer
Dim strsql As String

   strsql = ""
   For idx = txtSub.LBound To txtSub.UBound
      ' if the control is dirty then include it in the update string
      If txtSub(idx).IsDirty Then
         strsql = strsql & txtSub(idx).pBoundField & "=" & txtSub(idx).mGetDBText & ","
      End If
   Next idx

   For idx = cboSub.LBound To cboSub.UBound
      ' if the control is dirty then include it in the update string
      If cboSub(idx).IsDirty Then
         strsql = strsql & cboSub(idx).pBoundField & "=" & cboSub(idx).mGetDBText & ","
      End If
   Next idx

   If Right$(strsql, 1) = "," Then
      strsql = Left$(strsql, Len(strsql) - 1)
   End If

   CreateSubUpdateString = strsql

Exit_CreateSubUpdateString:
   Exit Function

Err_CreateSubUpdateString:
   ProcessDBError
   Resume Exit_CreateSubUpdateString

End Function

' Create the string that will be used for inserting records into the table
Private Function CreateSubInsertString(NewIndex As Long) As String
On Error GoTo Err_CreateSubInsertString

Dim idx As Integer
Dim strsql As String
Dim strfields As String
Dim strvals As String
```

```
    strfields = "InvNo,InvType,"
    strvals = m_pSelNode.SelNodeId & "," & INV_TYPE & ","

    For idx = txtSub.LBound To txtSub.UBound
        ' if the control is dirty, then include in the insert string
        If txtSub(idx).pBoundField <> "" Then
            strfields = strfields & txtSub(idx).pBoundField & ","
            strvals = strvals & txtSub(idx).mGetDBText & ","
        End If
    Next idx

    For idx = cboSub.LBound To cboSub.UBound
        ' if the control is dirty, then include in the insert string
        If cboSub(idx).pBoundField <> "" Then
            strfields = strfields & cboSub(idx).pBoundField & ","
            strvals = strvals & cboSub(idx).mGetDBText & ","
        End If
    Next idx

    If Right$(strfields, 1) = "," Then
        strfields = Left$(strfields, Len(strfields) - 1)
    End If

    If Right$(strvals, 1) = "," Then
        strvals = Left$(strvals, Len(strvals) - 1)
    End If

    If strfields <> "" And strvals <> "" Then
        CreateSubInsertString = "(" & strfields & ") values (" & strvals & ")"
    End If

Exit_CreateSubInsertString:
    Exit Function

Err_CreateSubInsertString:
    ProcessDBError
    Resume Exit_CreateSubInsertString

End Function

' Reset all the dirty controls to normal state
Private Sub ResetSubDirtyFlag()
On Error GoTo Err_ResetSubDirtyFlag

Dim idx As Integer

    For idx = txtSub.LBound To txtSub.UBound
        If txtSub(idx).IsDirty Then
            txtSub(idx).IsDirty = False
        End If
    Next idx

    For idx = cboSub.LBound To cboSub.UBound
        If cboSub(idx).IsDirty Then
            cboSub(idx).IsDirty = False
        End If
    Next idx

Exit_ResetSubDirtyFlag:
    Exit Sub

Err_ResetSubDirtyFlag:
    ProcessDBError
    Resume Exit_ResetSubDirtyFlag
End Sub

Private Sub AddNewSubRecord()
On Error GoTo Err_AddNewSubRecord
```

```
Dim newidx As Long

   If m_pSelNode.SelNodeId = -1 Or _
     pCurMode = MODE_ADD Then
      Exit Sub
   End If

   If SaveSuccess Then
      newidx = AddBlankItem
      lvwSub.SetSelectedItem newidx
      lvwSub.EnsureVisible newidx, False
      lvwSub.ActivateEditControls newidx, 0
      pCurMode = MODE_ADD
   End If

Exit_AddNewSubRecord:
   Exit Sub

Err_AddNewSubRecord:
   ProcessDBError
   Resume Exit_AddNewSubRecord
End Sub


Private Sub DeleteSubRecord()
On Error GoTo Err_DeleteSubRecord

Dim strsql As String
Dim RecCount As Long
Dim delIndex As Long
Dim idx As Long
Dim totItemCount As Long
Dim cData As clsGetData

   delIndex = lvwSub.GetSelectedItem
   totItemCount = lvwSub.Count

   If m_pSelNode.SelNodeId = -1 Or _
     totItemCount = 0 Or _
     lvwSub.GetSelectedItem = -1 Or _
     pCurMode = MODE_ADD Then
      Exit Sub
   End If

   If SaveSuccess Then
      If MsgBox("Do you really want to delete the selected Indent Item?", vbYesNo + vbQuestion +
vbDefaultButton2) = vbYes Then
         strsql = "delete from IndentItems " & _
                " where ItemSl=" & lvwSub.GetTag(delIndex, 1)

         Set cData = New clsGetData
         Call cData.RunQuery(m_ConStr, strsql, RecCount, CURRENT_USERID, CURRENT_COMPUTERNAME)

         If RecCount > 0 Then
            lvwSub.DeleteItem delIndex
         End If
      End If
      lvwSub.SetFocus
   End If

Exit_DeleteSubRecord:
   Set cData = Nothing
   Exit Sub

Err_DeleteSubRecord:
   ProcessDBError
   Resume Exit_DeleteSubRecord
End Sub
```

```
Private Sub UndoSubRecord()
On Error GoTo Err_UndoSubRecord

   lvwSub.SetFocus
   lvwSub.HideEditControls

   Select Case pCurMode
      Case MODE_ADD, MODE_ADD_AND_EDITING
         lvwSub.DeleteItem lvwSub.GetSelectedItem
   End Select

   pCurMode = MODE_VIEW

Exit_UndoSubRecord:
   Exit Sub

Err_UndoSubRecord:
   ProcessDBError
   Resume Exit_UndoSubRecord
End Sub


Private Sub InitOneFields()
On Error GoTo Err_InitOneFields

Dim cData As clsGetData
Dim strsql As String
Dim nextWo As Variant
Dim curDate As Date
Dim RecCount As Long
Dim recArray As Variant

   strsql = "select max(InvNo) From Indent "
   Set cData = New clsGetData
   nextWo = nz(cData.RunSQLReturnValue(m_ConStr, strsql), 0) + 1

   strsql = "select InvPrepBy, InvAppBy, InvSupBy from Warehouse"
   Call cData.GetRecordsFromSQL(m_ConStr, strsql, RecCount, recArray)

   m_pUserChanged = False

   txtFields(0) = nz(nextWo, "")
   txtFields(4) = INV_TYPE
   curDate = ConvertDateDMY(Date)
   txtFields(1) = curDate
   txtFields(2) = curDate
   txtFields(3) = curDate
   If RecCount = 1 Then
      cboFields(1) = nz(recArray(0, 0), 0)
      cboFields(2) = nz(recArray(1, 0), 0)
      cboFields(3) = nz(recArray(2, 0), 0)
   End If

   m_pUserChanged = True

Exit_InitOneFields:
   Set cData = Nothing
   Erase recArray
   Exit Sub

Err_InitOneFields:
   ProcessDBError
   Resume Exit_InitOneFields
End Sub

Private Sub Sub_Change()
On Error GoTo Err_Sub_Change
```

```
    If lvwSub.RaiseUpdate Then
        lvwSub.bEditing = True
        If pCurMode = MODE_VIEW Then
            pCurMode = MODE_VIEW_AND_EDITING
        ElseIf pCurMode = MODE_ADD Then
            pCurMode = MODE_ADD_AND_EDITING
        End If
        pSubDirty = True
    End If

Exit_Sub_Change:
    Exit Sub

Err_Sub_Change:
    ProcessDBError
    Resume Exit_Sub_Change
End Sub

Private Sub Sub_KeyPress(Index As Integer, KeyAscii As Integer)
On Error GoTo Err_Sub_KeyPress

    If KeyAscii = 10 Or KeyAscii = 13 Then
        SendKeys "{TAB}"
        KeyAscii = 0
    ElseIf KeyAscii = 27 Then
        UndoSubRecord
        KeyAscii = 0
    End If

Exit_Sub_KeyPress:
    Exit Sub

Err_Sub_KeyPress:
    ProcessDBError
    Resume Exit_Sub_KeyPress
End Sub

Private Sub Main_KeyPress(Index As Integer, KeyAscii As Integer, Optional pMultiline As Boolean = False)
On Error GoTo Err_Main_KeyPress

    If (KeyAscii = 10 Or KeyAscii = 13) And (Not pMultiline) Then
        SendKeys "{TAB}"
        KeyAscii = 0
    ElseIf KeyAscii = 27 Then
        UndoMainRecord
        KeyAscii = 0
    End If

Exit_Main_KeyPress:
    Exit Sub

Err_Main_KeyPress:
    ProcessDBError
    Resume Exit_Main_KeyPress
End Sub

Private Sub Main_Change()
On Error GoTo Err_Main_Change

    If m_pUserChanged Then
        If pCurMode = MODE_VIEW Then
            pCurMode = MODE_VIEW_AND_EDITING
        ElseIf pCurMode = MODE_ADD Then
            pCurMode = MODE_ADD_AND_EDITING
        End If
        pMainDirty = True
    End If

Exit_Main_Change:
```

```
    Exit Sub

Err_Main_Change:
    ProcessDBError
    Resume Exit_Main_Change
End Sub

Private Sub CalcSubTotFields()
On Error GoTo Err_CalcSubTotFields

Dim tmpC As Long
Dim tmpA As Double
Dim qty As Double
Dim idx As Long
Dim amt As Variant
Dim qtystr As String
Dim amtstr As String

    tmpC = lvwSub.Count
    tmpA = 0

    totFields(0) = tmpC

Exit_CalcSubTotFields:
    Exit Sub

Err_CalcSubTotFields:
    ProcessDBError
    Resume Exit_CalcSubTotFields

End Sub

Private Sub UpdateTotalNode()
    lblTotalNode = lblTotalNode.Tag & CStr(m_pRecCount)
End Sub

Private Sub GetNodeInfo(tNode As tagSelNodeOne, nodePtr As Long)
On Error GoTo Err_GetNodeInfo

Dim thNode As tagHSelNodeOne
Dim hitemptr As Long

    ' first get the hitem from node
    hitemptr = tvwMain.GetHItem(nodePtr)

    If hitemptr <> 0 Then
        CopyMemory thNode, ByVal hitemptr, Len(thNode)
        tNode.SelNodeId = thNode.SelNodeId
        tNode.SelNodeType = thNode.SelNodeType
        tNode.TypeOfNode = thNode.TypeOfNode
        tNode.SelTreeNodePtr = nodePtr
    Else
        tNode.SelNodeId = -1
        tNode.SelNodeType = ""
        tNode.TypeOfNode = 0
        tNode.SelTreeNodePtr = nodePtr
    End If

Exit_GetNodeInfo:
    Exit Sub

Err_GetNodeInfo:
    ProcessDBError
    Resume Exit_GetNodeInfo
End Sub

Private Sub FreeNodeInfo(tvhid As Long)
On Error GoTo Err_FreeNodeInfo
```

```
Dim tvid As tagHSelNodeOne
Dim hitemptr As Long

   ' after a node is deleted, clear the memory associated with it
   hitemptr = tvwMain.GetHItem(tvhid)
   If hitemptr <> 0 Then
      MoveMemory tvid, ByVal hitemptr, Len(tvid)
      isMalloc.Free ByVal hitemptr
  End If

Exit_FreeNodeInfo:
   Exit Sub

Err_FreeNodeInfo:
   ProcessDBError
   Resume Exit_FreeNodeInfo
End Sub

' Finds a specific node pointer from the nodes array. take a long
' value as parameter and compares with a field in hitem of every
' node in the node array
Private Function FindNodeInTree(recId As Long) As Long
On Error GoTo Err_FindNodeInTree

Dim idx As Long
Dim tvid As tagSelNodeOne

   FindNodeInTree = 0
   ' search through the node pointer array
   For idx = LBound(m_NodePtr, 1) To UBound(m_NodePtr, 1)
      ' if there is hitem associated
      ' get the item to tvid
      GetNodeInfo tvid, m_NodePtr(idx)
      ' does the search id maches with this node?
      If tvid.SelNodeId = recId Then
         ' if yes, then return the node id
         FindNodeInTree = m_NodePtr(idx)
         Exit For
      End If
    Next

Exit_FindNodeInTree:
   Exit Function

Err_FindNodeInTree:
   ProcessDBError
   Resume Exit_FindNodeInTree
End Function

Private Sub UpdateNode(nodePtr As Long, updateText As String)
On Error GoTo Err_UpdateNode

Dim thNode As tagHSelNodeOne
Dim hitemptr As Long

   ' first get the hitem from node
   hitemptr = tvwMain.GetHItem(nodePtr)

   If hitemptr <> 0 Then
      CopyMemory thNode, ByVal hitemptr, Len(thNode)
      thNode.SelNodeId = CLng(updateText)
      CopyMemory ByVal hitemptr, thNode, Len(thNode)
   End If

   tvwMain.SetNodeText nodePtr, updateText
   GetNodeInfo m_pSelNode, nodePtr

Exit_UpdateNode:
   Exit Sub
```

```
Err_UpdateNode:
   ProcessDBError
   Resume Exit_UpdateNode

End Sub


''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' This procedure fills the dropdown list of the form
' that will be used for data selection
' Last Update Date: April 2, 2003
'
' Parameters: none
' Return Value: none
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''

Private Sub FillFormCombo( _
   m_FacilityArray As Variant, _
   m_EmpArray As Variant, _
   m_IndDesigArray As Variant, _
   m_ItemsArray As Variant _
   )
On Error GoTo Err_FillFormCbo

Dim cData As clsGetData
Dim RecCount As Long
Dim strsql As String
Dim idx As Long

   Set cData = New clsGetData

   Call cData.GetRecordsFromSP(m_ConStr, "GetFacilityList", RecCount, m_FacilityArray)

   Call cData.GetRecordsFromSP(m_ConStr, "GetEmpList", RecCount, m_EmpArray)

   Call cData.GetRecordsFromSP(m_ConStr, "GetIndentDesigList", RecCount, m_IndDesigArray)

   Call cData.GetRecordsFromSP(m_ConStr, "GetLotPickListForIssue", RecCount, m_ItemsArray,
Array(mp("@SqlType", adInteger, 4, 0), mp("@GroupCode", adVarChar, 100, "(ALL)")))

Exit_FillFormCbo:
   Set cData = Nothing
   Exit Sub

Err_FillFormCbo:
   ProcessDBError
   Resume Exit_FillFormCbo
End Sub


Private Sub GetTreeWoList( _
   recArray As Variant, _
   RecCount As Long, _
   pQtype As Long, _
   Optional pSDate As String, _
   Optional pEDate As String, _
   Optional pInvNo As Long _
   )

Dim cData As clsGetData
Dim strsql As String

   Set cData = New clsGetData

   If pSDate = "" Then
      pSDate = "01/01/1900"
   Else
      pSDate = FormatDateADO(pSDate)
```

```
   End If
   If pEDate = "" Then
      pEDate = "01/01/2900"
   Else
      pEDate = FormatDateADO(pEDate)
   End If

         Call cData.GetRecordsFromSP(m_ConStr, "GetIssueInvTree", RecCount, recArray, _
               Array(mp("@SqlType", adInteger, 4, pQtype), _
               mp("@InvType", adInteger, 4, INV_TYPE), _
               mp("@pSDate", adVarChar, 10, pSDate), _
               mp("@pEDate", adVarChar, 10, pEDate), _
               mp("@InvNo", adInteger, 4, pInvNo)))

End Sub

Public Sub PopulateTreeMain( _
  tvwMain As MyTreeView, _
  recArray As Variant, _
  m_pRecCount As Long, _
  m_NodePtr() As Long, _
  Optional brefresh As Boolean, _
  Optional bGrouping As Boolean = False _
)
On Error GoTo Err_PopulateTreeMain

Dim lptvid As Long
Dim tvhid_one As tagHSelNodeOne
Dim lastclient As String
Dim lastdate As String
Dim lastdyeline As Long
Dim hitemRoot  As Long
Dim hitemchild As Long
Dim hitemClient As Long
Dim hitemParent As Long
Dim idx As Long
Dim tmpIdIndex As Long
Dim tmpNodeIndex As Long
Dim ntext As String
Dim curstdate As String

Dim zProgBar As CProgBar32
Dim zProgBarMax As Integer
Dim zProgBarStep  As Integer
Dim zIter As Integer
Dim pRect As RECT

   Set zProgBar = New CProgBar32
   ' initialize the progress bar
   With zProgBar
    .SethWndParent = frmMain.sBar.GetStatBarHwnd  'Me.hwnd
    frmMain.sBar.GetPanelRect "Panel_1", pRect.Left, pRect.Top, pRect.Right, pRect.Bottom
    .Create pRect.Left, pRect.Top, pRect.Right - pRect.Left, pRect.Bottom - pRect.Top, False
   End With

   ' clear the tree first
   tvwMain.DeleteAllItems

   ' insert root node
   hitemRoot = tvwMain.AddNode(, ROOT_TEXT, m_pRecCount <> 0, , ROOT_NODE_ICON, ROOT_NODE_ICON)

   If m_pRecCount <> 0 Then
      zProgBarMax = m_pRecCount
      zProgBar.setrange 0, zProgBarMax
      zProgBarStep = zProgBarMax \ 100
      If zProgBarStep = 0 Then
         zProgBarStep = m_pRecCount
      End If
      zProgBar.SetStep zProgBarStep
```

```
    End If

    ' create the node pointer array
    ReDim m_NodePtr(0 To m_pRecCount)
    m_NodePtr(0) = hitemRoot

   lastclient = -999
   lastdate = ""
   lastdyeline = -999
   ' create all the nodes
   For idx = 1 To m_pRecCount
       If Not bGrouping Then
          tvhid_one.SelNodeId = recArray(0, idx - 1)
          tvhid_one.SelNodeType = -1
          tvhid_one.TypeOfNode = 1
          lptvid = SetNodeInfo_One(tvhid_one)
          hitemchild = tvwMain.AddNode(hitemRoot, recArray(0, idx - 1), , lptvid, LEAF_NODE_ICON,
LEAF_NODE_SELICON)
       Else
          If lastclient <> recArray(0, idx - 1) Then
             hitemClient = tvwMain.AddNode(hitemRoot, recArray(0, idx - 1), True, , PARENT_NODE_ICON,
PARENT_NODE_SELICON)
             lastclient = recArray(0, idx - 1)
          End If
          tvhid_one.SelNodeId = recArray(1, idx - 1)
          tvhid_one.SelNodeType = -1
          tvhid_one.TypeOfNode = 1
          lptvid = SetNodeInfo_One(tvhid_one)
          hitemchild = tvwMain.AddNode(hitemClient, recArray(1, idx - 1), , lptvid, LEAF_NODE_ICON,
LEAF_NODE_SELICON)
       End If

       ' store node id and hitem id in node pointer array
       m_NodePtr(idx) = hitemchild

       If idx Mod zProgBarStep = 0 Then
          zProgBar.StepIt
       End If

    Next idx

    ' Expand the root
    tvwMain.Expand hitemRoot, TVE_EXPAND

    Call UpdateWindow(tvwMain.hwnd)

    If brefresh Then
       ' find the first child
       hitemchild = tvwMain.GetChild(hitemRoot)
       If hitemchild = 0 Then
          ' if no child, then select the root
          tvwMain.SelectItem (hitemRoot)
       Else
          ' else, select the first child
          tvwMain.SelectItem hitemchild
       End If
    End If

Exit_PopulateTreeMain:
    zProgBar.DestroyProgBar
    Exit Sub

Err_PopulateTreeMain:
    ProcessDBError
    Resume Exit_PopulateTreeMain
End Sub
```

```
Private Sub GetOneWoInfo(ByVal pInvNo As Long, oneCount As Long, oneArray As Variant, Optional manyCount As
Long, Optional manyArray As Variant)
Dim cData As clsGetData
Dim strsql As String

   Set cData = New clsGetData

   Call cData.GetRecordsFromSP(m_ConStr, "GetIssueInvOneInfo", oneCount, oneArray, _
            Array(mp("@SqlType", adInteger, 4, 1), mp("@InvType", adInteger, 4, INV_TYPE), mp("@InvNo",
adInteger, 4, m_pSelNode.SelNodeId)))

   If Not IsMissing(manyArray) Then
      Call cData.GetRecordsFromSP(m_ConStr, "GetIssueInvManyInfo", manyCount, manyArray, _
            Array(mp("@SqlType", adInteger, 4, 1), mp("@InvType", adInteger, 4, INV_TYPE), mp("@InvNo",
adInteger, 4, m_pSelNode.SelNodeId)))
   End If

   Set cData = Nothing
End Sub

Private Function PostToStock() As Boolean
On Error GoTo Err_PostToStock

Dim cData As clsGetData
Dim strsql As String
Dim RecCount As Long

   PostToStock = False

   If m_pSelNode.SelNodeId <> -1 And lvwSub.Count > 0 Then
      If MsgBox("Do you really want to update " & lvwSub.Count & " items to stock?" & vbCrLf & _
            "You will not be able to edit this invoice after update", vbYesNo + vbQuestion + vbDefaultButton2) =
vbYes Then
         Set cData = New clsGetData

         RecCount = cData.RunSPReturnInteger(m_ConStr, "PostToStock_IssueInv", Array(mp("@InvType",
adInteger, 4, INV_TYPE), mp("@InvNo", adInteger, 4, m_pSelNode.SelNodeId)))
         If RecCount > 0 Then
               MsgBox lvwSub.Count & " items posted to stock successfully", vbInformation
               m_pUserChanged = False
               SetCheckValue chkFields(0).hwnd, vbChecked
               m_pUserChanged = True
               cmdMain(6).Enabled = False
               cmdMain(3).Enabled = False
               PostToStock = True
         Else
               MsgBox "Cannot post more than Stock Quantity", vbInformation
         End If
      End If
   End If

Exit_PostToStock:
   Set cData = Nothing
   Exit Function

Err_PostToStock:
   ProcessDBError
   Resume Exit_PostToStock
End Function

Private Sub LockFormEditing()
On Error GoTo Err_LockFormEditing

Dim idx As Long

   For idx = txtFields.LBound To txtFields.UBound
      txtFields(idx).Locked = True
   Next idx
```

```
    For idx = cboFields.LBound To cboFields.UBound
       cboFields(idx).Enabled = False
    Next idx

    For idx = chkFields.LBound To chkFields.UBound
       chkFields(idx).Enabled = False
    Next idx

    lvwSub.EditEnabled = False

    For idx = cmdSub.LBound To cmdSub.UBound
       cmdSub(idx).Enabled = False
    Next idx

    cmdMain(0).Enabled = False
    cmdMain(1).Enabled = False
    cmdMain(2).Enabled = False
    cmdMain(3).Enabled = False
    cmdMain(6).Enabled = False

    pCurMode = MODE_READONLY

Exit_LockFormEditing:
    Exit Sub

Err_LockFormEditing:
    ProcessDBError
    Resume Exit_LockFormEditing
End Sub

Private Sub ICrystalEngine_getParameters(ByVal jobHandle As Integer)
    ParamValInfo.StructSize = PE_SIZEOF_VALUE_INFO
    ParamValInfo.ValueType = PE_VI_STRING
    ParamValInfo.viString = GetWarehouseForReport & Chr$(0)
    Call PEAddParameterCurrentValue(jobHandle, "FacilityHeader" & Chr$(0), "" + Chr$(0), ParamValInfo)

    ParamValInfo.StructSize = PE_SIZEOF_VALUE_INFO
    ParamValInfo.ValueType = PE_VI_NUMBER
    ParamValInfo.viNumber = INV_TYPE
    Call PEAddParameterCurrentValue(jobHandle, "@InvType" & Chr$(0), "" + Chr$(0), ParamValInfo)

    ParamValInfo.ValueType = PE_VI_NUMBER
    ParamValInfo.viNumber = m_pSelNode.SelNodeId
    Call PEAddParameterCurrentValue(jobHandle, "@InvNo" & Chr$(0), "" + Chr$(0), ParamValInfo)

    ParamValInfo.StructSize = PE_SIZEOF_VALUE_INFO
    ParamValInfo.ValueType = PE_VI_NUMBER
    ParamValInfo.viNumber = m_pCurrentCopy
    Call PEAddParameterCurrentValue(jobHandle, "pCopy" & Chr$(0), "" + Chr$(0), ParamValInfo)

End Sub

Private Function RecordIsValid() As Boolean

Dim pInvNo As String
Dim pIndentor As Long
Dim pIndentNo As String
Dim pPrepDate As String
Dim pPrepBy As Long
Dim pErrorMsg As String
Dim pFocusGiven As Boolean
Dim pAppDate As String
Dim pAppBy As Long
Dim pSupDate As String
Dim pSupBy As Long

    RecordIsValid = False

    pErrorMsg = ""
```

```
pInvNo = Trim(txtFields(0))
pIndentor = cboFields(0).ListIndex
pPrepBy = cboFields(1).ListIndex
pPrepDate = Trim(txtFields(1))
pAppBy = cboFields(2).ListIndex
pAppDate = Trim(txtFields(2))
pSupBy = cboFields(3).ListIndex
pSupDate = Trim(txtFields(3))

If (pInvNo = "") Then
   pErrorMsg = pErrorMsg & "'Invoice No.', "
   If Not pFocusGiven Then
      Set m_pCurMainCtl = txtFields(0)
      pFocusGiven = True
   End If
End If
If (pIndentor = -1) Then
   pErrorMsg = pErrorMsg & "'Indentor', "
   If Not pFocusGiven Then
      Set m_pCurMainCtl = cboFields(0)
      pFocusGiven = True
   End If
End If
If (pPrepBy = -1) Then
   pErrorMsg = pErrorMsg & "'Prepared By', "
   If Not pFocusGiven Then
      Set m_pCurMainCtl = cboFields(1)
      pFocusGiven = True
   End If
End If
If (pPrepDate = "") Then
   pErrorMsg = pErrorMsg & "'Preparation Date', "
   If Not pFocusGiven Then
      Set m_pCurMainCtl = txtFields(1)
      pFocusGiven = True
   End If
End If
If (pAppBy = -1) Then
   pErrorMsg = pErrorMsg & "'Approved By', "
   If Not pFocusGiven Then
      Set m_pCurMainCtl = cboFields(2)
      pFocusGiven = True
   End If
End If
If (pAppDate = "") Then
   pErrorMsg = pErrorMsg & "'Approved Date', "
   If Not pFocusGiven Then
      Set m_pCurMainCtl = txtFields(2)
      pFocusGiven = True
   End If
End If
If (pSupBy = -1) Then
   pErrorMsg = pErrorMsg & "'Supplied By', "
   If Not pFocusGiven Then
      Set m_pCurMainCtl = cboFields(3)
      pFocusGiven = True
   End If
End If
If (pSupDate = "") Then
   pErrorMsg = pErrorMsg & "'Supplied Date', "
   If Not pFocusGiven Then
      Set m_pCurMainCtl = txtFields(3)
      pFocusGiven = True
   End If
End If

If pErrorMsg <> "" Then
   pErrorMsg = Left$(pErrorMsg, Len(pErrorMsg) - 2)
   pErrorMsg = pErrorMsg & " cannot be Empty"
```

```
      MsgBox pErrorMsg, vbCritical, "ERROR"
   Else
      RecordIsValid = True
   End If

End Function


Private Function SubRecordIsValid() As Boolean

Dim pItem As Long
Dim pReqQty As String
Dim pIssueQty As String
Dim pReason As Long
Dim pErrorMsg As String
Dim pFocusGiven As Boolean

   SubRecordIsValid = False

   pErrorMsg = ""
   pItem = cboSub(0).ListIndex
   pIssueQty = Trim(txtSub(1))

   If (pItem = -1) Then
      pErrorMsg = pErrorMsg & "'Item Name', "
      If Not pFocusGiven Then
         Set m_pCurMainCtl = cboSub(0)
         pFocusGiven = True
      End If
   End If

   If pErrorMsg <> "" Then
      pErrorMsg = Left$(pErrorMsg, Len(pErrorMsg) - 2)
      pErrorMsg = pErrorMsg & " cannot be Empty"
   End If

   If (pIssueQty <> "" And (Not IsNumeric(pIssueQty))) Then
      If pErrorMsg <> "" Then
         pErrorMsg = pErrorMsg & vbCrLf
      End If
      pErrorMsg = pErrorMsg & "'Issued Qty' is not a Valid Value"
      If Not pFocusGiven Then
         Set m_pCurMainCtl = txtSub(1)
         pFocusGiven = True
      End If
   End If

   If (pIssueQty = "") Then
      If pErrorMsg <> "" Then
         pErrorMsg = pErrorMsg & vbCrLf
      End If
      pErrorMsg = pErrorMsg & "'Issued Qty' cannot be Empty"
      If Not pFocusGiven Then
         Set m_pCurMainCtl = txtSub(1)
         pFocusGiven = True
      End If
   End If

   If pErrorMsg <> "" Then
      MsgBox pErrorMsg, vbCritical, "ERROR"
   Else
      SubRecordIsValid = True
   End If

End Function


Private Function CanSaveSubRecord() As Boolean
```

```
On Error GoTo Err_CanSaveSubRecord

Dim pItemCode As String
Dim pItemQty As Double
Dim pLotNo As String
Dim pMfgDate As String
Dim pExpDate As String
Dim pErrorMsg As String
Dim pFocusGiven As Boolean
Dim cData As clsGetData
Dim RecCount As Long
Dim recArray() As Variant

   CanSaveSubRecord = False

   Set cData = New clsGetData

   pErrorMsg = ""
   pItemCode = Trim(cboSub(0))
   pItemQty = CDbl(Trim(txtSub(1)))
   pLotNo = Trim(txtSub(2))
   pMfgDate = FormatDateADO(Trim(txtSub(3)))
   pExpDate = FormatDateADO(Trim(txtSub(4)))

   Call cData.GetRecordsFromSP(m_ConStr, "GetItemCurStockQty", _
            RecCount, recArray, Array(mp("@ItemCode", adVarChar, 100, pItemCode), mp("@LotNo", adVarChar,
100, pLotNo), mp("@MfgDate", adVarChar, 100, pMfgDate), mp("@ExpDate", adVarChar, 100, pExpDate)))
   If RecCount > 0 Then
      If pItemQty > CDbl(recArray(0, 0)) Then
         pErrorMsg = pErrorMsg & "Cannot Issue More Than Stock Quantity." & vbCrLf & _
                  "Current Stock Quantity: " & Format(CDbl(recArray(0, 0)), "#,###")
         If Not pFocusGiven Then
            Set m_pCurMainCtl = txtSub(1)
            pFocusGiven = True
         End If
      End If
   End If

   If pErrorMsg <> "" Then
      MsgBox pErrorMsg, vbCritical, "ERROR"
   Else
      CanSaveSubRecord = True
   End If

Exit_CanSaveSubRecord:
   Set cData = Nothing
   Erase recArray
   Exit Function

Err_CanSaveSubRecord:
   CanSaveSubRecord = False
   Resume Exit_CanSaveSubRecord
End Function


Private Sub cmdClose_Click()
   picPrintOptions.Visible = False
End Sub

Private Sub cmdPrint_Click()
Dim idx As Integer
Dim bPosted As Boolean

   If chkFields(0).Value = vbUnchecked Then
      bPosted = PostToStock
   Else
      bPosted = True
   End If
```

```
    If bPosted Then
        For idx = optPrint.LBound To optPrint.UBound
            If optPrint(idx).Value Then
                PreviewRecord idx
                Exit For
            End If
        Next idx
    End If
    cmdClose_Click
End Sub

'========  END of USER-DEFINED FUNCTIONS ========================='

'======= START of LISTVIEW INTERFACE IMPLEMENTATION  ====================='

Private Sub IListViewOwner_AfterEditComplete(hwndLVW As Long)
'
End Sub

Private Sub IListViewOwner_AfterItemClick(hwndLVW As Long, ItemIndex As Long, subitemIndex As Long)
'
End Sub

Private Sub IListViewOwner_AfterUpdate(hwndLVW As Long, ItemIndex As Long)
On Error GoTo Err_IListViewOwner_AfterUpdate

    If lvwSub.hwnd = hwndLVW Then
        ' recalc the total fields
        CalcSubTotFields
    End If

Exit_IListViewOwner_AfterUpdate:
    Exit Sub

Err_IListViewOwner_AfterUpdate:
    ProcessDBError
    Resume Exit_IListViewOwner_AfterUpdate
End Sub

Private Sub IListViewOwner_BeforeListViewClick(hwndLVW As Long, isOK As Boolean, ItemIndex As Long)
On Error GoTo Err_IListViewOwner_BeforeListViewClick

    If lvwSub.hwnd = hwndLVW Then
        ' return the save status before clicking on the listview
        isOK = SaveSuccess
    End If

Exit_IListViewOwner_BeforeListViewClick:
    Exit Sub

Err_IListViewOwner_BeforeListViewClick:
    ProcessDBError
    Resume Exit_IListViewOwner_BeforeListViewClick
End Sub

Private Sub IListViewOwner_BeforeUpdate(hwndLVW As Long, isOK As Boolean, ItemIndex As Long)
On Error GoTo Err_IListViewOwner_BeforeUpdate

    If lvwSub.hwnd = hwndLVW Then
        If ItemIndex <> -1 Then
            Select Case pCurMode
                Case MODE_VIEW_AND_EDITING, MODE_ADD_AND_EDITING
                    isOK = SaveSubRecord(ItemIndex)

                Case MODE_ADD
                    isOK = False

                Case MODE_VIEW
                    isOK = True
```

```
            End Select
        Else
            isOK = True
        End If
    End If

Exit_IListViewOwner_BeforeUpdate:
    Exit Sub

Err_IListViewOwner_BeforeUpdate:
    ProcessDBError
    Resume Exit_IListViewOwner_BeforeUpdate
End Sub

Private Sub IListViewOwner_AfterItemDblClick(hwndLVW As Long, ItemIndex As Long, subitemIndex As Long)
'
End Sub

Private Sub IListViewOwner_AfterItemSelected(hwndLVW As Long, ItemIndex As Long)
'
End Sub

Private Sub IListViewOwner_CustomDraw(hwndLVW As Long, iItem As Long, iSubItem As Long, TextBkColor As
Long, TextForeColor As Long, bBold As Boolean)
'
End Sub

Private Sub IListViewOwner_CustomDrawIcon(hwndLVW As Long, iItem As Long, hdc As Long, rcLeft As Long,
rcTop As Long, rcRight As Long, rcBottom As Long, bSelState As Boolean)
'
End Sub

'======== END of LISTVIEW INTERFACE IMPLEMENTATION  ====================='

'======== START of TREEVIEW INTERFACE IMPLEMENTATION  ====================='

Private Sub ITreeViewOwner_AfterNodeDelete(nodePtr As Long)
On Error GoTo Err_tvwMain_AfterNodeDelete

    FreeNodeInfo nodePtr

Exit_tvwMain_AfterNodeDelete:
    Exit Sub

Err_tvwMain_AfterNodeDelete:
    ProcessDBError
    Resume Exit_tvwMain_AfterNodeDelete
End Sub

Private Sub ITreeViewOwner_AfterSelectionChange(nodePtr As Long)
On Error GoTo Err_tvwMain_AfterSelectionChange

Dim oneCount As Long
Dim manyCount As Long
Dim oneArray() As Variant
Dim manyArray() As Variant

    ' select the first tab
    If tabMain.SelectedTab <> 1 Then
        tabMain.SelectTab BASIC_TAB
        tabMain_TabClick 1
    End If

    GetNodeInfo m_pSelNode, nodePtr

    If m_pSelNode.TypeOfNode = 0 Then
        ' if the node has children then clear everything
        ClearControls
        CalcSubTotFields
```

```
    Else
        Call GetOneWoInfo(m_pSelNode.SelNodeId, oneCount, oneArray, manyCount, manyArray)
        ' show basic information
        FillOneInfo oneArray, oneCount
        FillManyInfo manyArray, manyCount

    End If
    cmdMain(3).Enabled = (m_pSelNode.TypeOfNode > 0) And (m_pCurMode = MODE_VIEW) And
(chkFields(0).Value = vbUnchecked)
    cmdMain(4).Enabled = (m_pSelNode.TypeOfNode > 0)
    cmdMain(6).Enabled = (m_pSelNode.TypeOfNode > 0) And (chkFields(0).Value = vbUnchecked)

Exit_tvwMain_AfterSelectionChange:
    Erase oneArray
    Erase manyArray
    Exit Sub

Err_tvwMain_AfterSelectionChange:
    ProcessDBError
    Resume Exit_tvwMain_AfterSelectionChange
End Sub

Private Sub ITreeViewOwner_BeforeNodeClick(isOK As Boolean, tItem As Long)
On Error GoTo Err_tvwMain_BeforeNodeClick

Dim oneArray() As Variant
Dim oneCount As Long

    ' user clicked on node, so try to save first
    If pMainDirty Then
        isOK = SaveSuccess
    ElseIf pSubDirty Then
        lvwSub.UpdateLVWStatus 0, 0, 0, 0, isOK
        Exit Sub
    End If

    ' if adding a new record, then just reset the flag
    If pCurMode = MODE_ADD Then
        pCurMode = MODE_VIEW
        If tItem = m_pSelNode.SelTreeNodePtr Then
            Call GetOneWoInfo(m_pSelNode.SelNodeId, oneCount, oneArray)
            FillOneInfo oneArray, oneCount
        End If
    End If

Exit_tvwMain_BeforeNodeClick:
    Erase oneArray
    Exit Sub

Err_tvwMain_BeforeNodeClick:
    ProcessDBError
    Resume Exit_tvwMain_BeforeNodeClick

End Sub

Private Sub ITreeViewOwner_BeforeSelectionChange(isOK As Boolean)
    ' before changing the node selection, to save first
    isOK = SaveSuccess
    ' if adding a new record, then just reset the flag
    If pCurMode = MODE_ADD Then
        pCurMode = MODE_VIEW
    End If
End Sub

'======= END of TREEVIEW INTERFACE IMPLEMENTATION  ====================='

'======= START of OWNERDRAW BUTTON INTERFACE IMPLEMENTATION  ====================='

Private Property Get IOwnerDrawButton_ButtonContainerhWnd() As Long
```

```
    IOwnerDrawButton_ButtonContainerhWnd = Me.frameButtons.hwnd
End Property

Private Property Get IOwnerDrawButton_DoOwnerDraw(ByVal lHwnd As Long) As Boolean
    IOwnerDrawButton_DoOwnerDraw = True
End Property

Private Sub IOwnerDrawButton_DrawItem(ByVal lHwnd As Long, ByVal lHDC As Long, ByRef lLeft As Long, ByRef
lTop As Long, ByRef lRight As Long, ByRef lBottom As Long, ByVal bPushed As Boolean, ByVal bChecked As
Boolean, ByVal bEnabled As Boolean, ByVal bInFocus As Boolean, ByRef bDoDefault As Boolean)
On Error GoTo Err_IOwnerDrawButton_DrawItem

Dim ImgIndex As Long
Dim cmdCaption As String

    Select Case lHwnd
     Case cmdMain(0).hwnd
        ImgIndex = 0
        cmdCaption = cmdMain(0).Caption

     Case cmdMain(1).hwnd
        ImgIndex = 1
        cmdCaption = cmdMain(1).Caption

     Case cmdMain(2).hwnd
        ImgIndex = 2
        cmdCaption = cmdMain(2).Caption

     Case cmdMain(3).hwnd
        ImgIndex = 3
        cmdCaption = cmdMain(3).Caption

     Case cmdMain(4).hwnd
        ImgIndex = 16
        cmdCaption = cmdMain(4).Caption

     Case cmdMain(5).hwnd
        ImgIndex = 5
        cmdCaption = cmdMain(5).Caption

     Case cmdMain(6).hwnd
        ImgIndex = 23
        cmdCaption = cmdMain(6).Caption

     Case cmdSearch.hwnd
        ImgIndex = 6
        cmdCaption = cmdSearch.Caption

     Case cmdFilter.hwnd
        ImgIndex = 7
        cmdCaption = cmdFilter.Caption

     Case cmdSub(0).hwnd
        ImgIndex = 0
        cmdCaption = ""

     Case cmdSub(1).hwnd
        ImgIndex = 3
        cmdCaption = ""

    End Select


    DrawButton lHwnd, lHDC, lLeft, lTop, lRight, lBottom, _
            m_pImageBtn.hIml, ImgIndex, m_pImageBtn.IconSizeX, cmdCaption, _
            bPushed, bEnabled, bInFocus


Exit_IOwnerDrawButton_DrawItem:
```

```
    Exit Sub

Err_IOwnerDrawButton_DrawItem:
    ProcessDBError
    Resume Exit_IOwnerDrawButton_DrawItem
End Sub


'======= END of OWNERDRAW BUTTON INTERFACE IMPLEMENTATION  ====================='
```

For more information, please visit deliver.jsi.com.

USAID | DELIVER PROJECT

John Snow, Inc.

1616 Fort Myer Drive, 11th Floor

Arlington, VA 22209 USA

Phone: 703-528-7474

Fax: 703-528-7480

Email: askdeliver@jsi.com

Internet: deliver.jsi.com