USAID | DELIVER PROJECT

FROM THE AMERICAN PEOPLE

PipeLine 5.1

Technical Documentation

PipeLine 5.0

Technical Documentation

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
Blankenship, Lisa. 2011. *PipeLine 5.0: Technical Documentation.* Arlington, Va.: USAID | DELIVER PROJECT, Task Order 1.

Abstract
Develop technical documentation for PipeLine so that the application is well understood by technical persons, easily maintainable and transferrable.   This will ensure that this application's software development life cycle (SDLC) be managed effectively in future.

Cover Photo: Woman enters shipment data in Zimbabwe, on November 16, 2009.

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

Contents

Figures

Acronyms

ADO	ActiveX Data Object
AIDS	Acquired Immune Deficiency Syndrome
DLL	Dynamic Link Library
ERD	Entity Relationship Diagram
FK	Foreign Key
NGO	Non-Governmental Organization
PK	Primary Key
SCMgr	Supply Chain Manager
SDG	Software Development Group
USAID	U.S. Agency for International Development
VB	Visual Basic
VBA	Visual Basic for Applications

Acknowledgments

The author would like to thank Jeffrey Leiner, Muhammad Ahmed, Ashraf Islam, Trisha Long, and Joy Kamunyori for their helpful suggestions and expert reviews of this document and the developers who worked on PipeLine and took the time to actually document the code making this effort easier (Muhammad Ahmed, Lisa Blankenship, Jeff Leiner, and Mike Tracy).

The author would also like to Acknowledge USAID for funding this effort and those from FPLM and DELIVER projects over the years who championed PipeLine development such as Trisha Long, Karen Ampeh, John Durgavich and the countless others who contributed in so many ways.

Introduction

Overview

The Pipeline Monitoring and Procurement Planning System (PipeLine), a software tool, was designed to help program managers monitor the status of their product pipelines and product procurement plans. PipeLine provides information needed to initiate and follow-up actions to ensure the regular and consistent stock of products at the program or national level. Consistency of stock is the first step in meeting the basic objective of any logistics system, which is to provide—

- the *right* quantities
- of the *right* commodities
- in the *right* condition
- in the *right* place
- at the *right* time
- for the *right* cost.

These are the *six rights* of logistics management.

What PipeLine Can Do for You

PipeLine helps you achieve the *right* quantities at the *right* time.

For each product, PipeLine monitors—

- Total quantities *consumed* (i.e., amounts dispensed to users or sold to clients)
- *Shipments* of new products (planned, ordered, shipped, or received) into your program
- and the values of your products
- *Inventory levels* for each product in your program's logistics system (desired and actual)
- *Inventory level changes* (e.g., product losses or transfers out of or into your program)

With these data and an understanding of the *lead time* required for each step in the procurement process, PipeLine can—

1. Show what actions you need to take for procurement planning and management, and when these actions should be taken.
2. Identify impending problems (i.e., surpluses, shortfalls, or stockouts) *before* they occur.
3. Calculate procurement quantities needed to keep your pipeline in balance.
4. Calculate the estimated value of shipments or maintain the actual value (if known).

You can use this information with program policymakers, product suppliers, and donors to provide a rational basis for planning future product needs.

PipeLine is *not* the answer to every logistics question. It helps monitor the *aggregate* quantity of each product entering and leaving your program's distribution system (preferably using data from a logistics management information system [LMIS]).

PipeLine's utility is enhanced if your program has a well-functioning LMIS and forecasting procedures. Even without these underlying systems, use PipeLine with whatever data are available. By beginning a rational and systematic product monitoring and planning process, you take the first step toward ensuring consistent stock levels.

Why Use PipeLine?

Ensuring adequate supplies of commodities is difficult for most programs. As a program manager, you face a complex procurement planning environment, characterized by—

1. Multiple suppliers of many products (local and private suppliers, bilateral and multilateral donors, etc.), each with its own products, lead times, costs, information needs, and bureaucratic constraints
2. Proliferation of service delivery points, in many cases in an integrated service delivery setting, and/or with multiple service delivery organizations served by a single logistics system
3. Increasing volume (and costs) of commodities, which must be managed and moved through complex distribution channels
4. Increasing emphasis on accountability, cost-effectiveness, and sustainability from donors who fund product procurement and from policymakers.

You need to monitor the quantity and timing of multiple products entering your logistics system from multiple suppliers. Because procurement lead times may be long—years, in many cases— you need to take action months or years before commodities are needed to receive them on time.

You may need to negotiate with many different suppliers and donors to obtain the quantities you require. Such negotiations are best accomplished when specific data on product requirements are available. You must know when you will stock out of each product, how much must be procured to meet future needs, and when you should receive it. To prevent overordering, you must also know what quantities would exceed your storage capacity or risk wastage due to expiry. PipeLine can provide this information.

Who Should Use PipeLine?

In a multi-product, multi-supplier environment, procurement planning and pipeline monitoring functions cannot be donor driven. It is increasingly necessary that local program managers be empowered to do their own forecasting, pipeline monitoring, and procurement planning; they must also take charge of coordinating the activities of donors and local suppliers, as well as those of their own logistics management staff. Donor staff often have other priorities and little time to devote to the details of logistics management. Commercial suppliers have interests that may or may not correspond to the interests of your organization.

If you are the logistics manager or program manager for your organization, you should manage your own pipeline. PipeLine can help.

While your managers and decision makers will be the primary users of PipeLine, the system can provide information to—

Suppliers of commodities
PipeLine provides reports on the current status and the cost of pending shipments from aspecific supplier, which that supplier can use to monitor product flow.

Purchasers/donors of commodities
Staff who finance the purchase of commodities can use PipeLine reports and graphs to understand the current pipeline status and future requirements.

Host-country policymakers
PipeLine reports and graphs can be used to help policymakers understand issues with the levels of particular commodities and the implications of different decisions on the availability of the product.

PipeLine Software Functions

PipeLine can help you with pipeline monitoring and procurement planning functions, as explained below.

Pipeline Monitoring

Pipeline monitoring functions include—

- Monitoring stock balances, in terms of quantities and months of stock on hand in the entire program (aggregate of stock at all levels)
- Comparing stock balances to maximum and minimum stock policies
- Automating the identification of pipeline problems (quantities needed, stockouts, balances below minimum or above maximum)
- Providing couple-years of protection (CYP) conversion graphs.

Procurement Planning

Procurement planning functions include—

- Calculation of shortfalls/surpluses and quantities needed to maintain the program's desired stock levels
- Automated calculation and tracking of pending pipeline actions, based on lead times (shipments to plan, order, ship, and receive)
- Application of USAID's contraceptive procurement tables (CPT) format for the computation of calendar year quantities required and the generation of data for USAID's planning requirements
- Calculation of estimated costs of shipments and freight
- Comparison of alternative procurement scenarios and analysis
- Alternative unit of measure calculation displays products in Basic Units. Basic Units are used to quantify patient or consumer needs and usually refers to tablets, capsules, or milliliters, rather than packs or bottles.

Technical Architecture

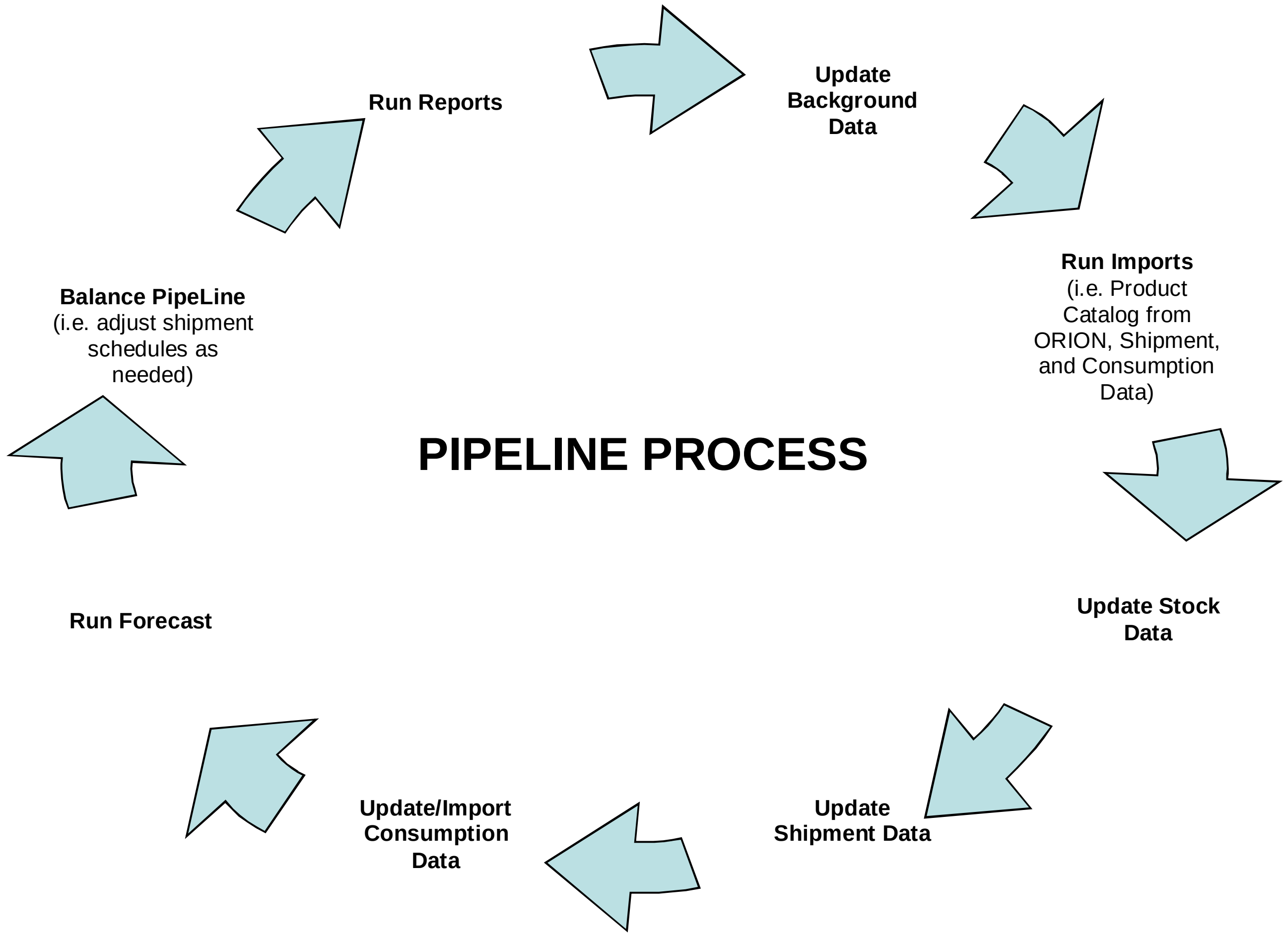
The PipeLine software is developed using following programming language, tools and techniques:

- Microsoft Access 2003 and 2000. The front end database is based on Access 2003 and the back-end database is based on Access 2000.
- On-line help was developed using Robohelp X5

- Automated installable version was created using InstallShield 12 and SageKey for Access 2003 version 2.0.9
- Data export/import interface between Supply Chain Manager and PipeLine is build using XML 1.1
- CD auto-run was build using AutoRun Pro Enterprise II
- User Guide and Technical documentation was developed using combination of Microsoft Visio, Powerpoint and Word
- For source control, Microsoft Visual Source Safe was used.
- For bug tracking and issue tracking, open source bugzilla application was used.

Process Flows

Figure 1 - PipeLine Process Flow



Run Reports
Update Background Data
Run Imports (i.e. Product Catalog from ORION, Shipment, and Consumption Data)
PIPELINE PROCESS
Update Stock Data
Update Shipment Data
Update/Import Consumption Data
Run Forecast
Balance PipeLine (i.e. adjust shipment schedules as needed)

Appendices

Table 1 - List of Appendices

Appendix	Title	Description
A	The Reddick VBA (RVBA) Naming Conventions, Version 6.01	Industry standard naming conventions for Access applications
B	The Reddick VBA (RVBA) Coding Conventions (version 0.90)	Industry standard coding conventions for Access applications
C	Microsoft Application User Interface Guidelines	JSI GUI standards

System Requirements

The following resources are recommended for use with PipeLine—

Table 2 - Hardware and Software Requirements

Component	Hardware/Software Requirements
CPU	Pentium IV or higher
Operating System	Windows XP or above
Memory	1 GB or higher
Hard Disk Space	500 MB of free space or higher
Video Adapter	SVGA with at least 800 X 600 resolution
Microsoft Office	Microsoft Office 2003 or higher

Installation and Configuration

Installation Instructions

How to Install PipeLine

PipeLine can be installed from a CD-ROM or the Internet.

Microsoft Office® 2003
Although PipeLine will run without Microsoft Office® 2003 installed on your computer, having Office 2003 installed will enhance PipeLine's usefulness by allowing PipeLine to export data files to Word® or Excel®.

Before You Begin

You can run PipeLine 2 and PipeLine 5. on the same computer, but we recommend that you uninstall PipeLine 2 BEFORE installing PipeLine 5.

To uninstall PipeLine 2—

1. Click on Start.
2. Click on the Settings option.
3. Click on the Control Panel option.

Previous Versions
All other previous versions of PipeLine cannot be run on the same computer as PipeLine 5. When you install PipeLine 5, the installer will automatically remove the previous version of PipeLine prior to installing PipeLine 5.

After the Control Panel window opens—

4. Click on the Add/Remove Programs option.

Locate and click on PipeLine in the Currently Installed Programs list.

5. Click on the Change/Remove button.

The PipeLine 2 setup program will start, and will prepare your computer to uninstall PipeLine 2.

6. Click on the Remove All Button.

A message is displayed asking if you want to remove PipeLine.

7. Click on the Yes button to begin the uninstall procedure.

When the uninstall procedure is completed, you will be prompted to restart windows.

8. Click on the Restart Windows button.

Data Files
Data files created with previous versions of PipeLine are not removed from your system. See Converting Your Existing Data Files on page 29 for information on converting your existing PipeLine 2 data files.

Installing PipeLine from a CD

1. Start Microsoft Windows.
2. Insert the PipeLine CD.

The PipeLine installation should begin automatically.

3. Follow the on-screen instructions.
4. If the installation does not begin automatically—
    a. Click on Start on the Windows Taskbar.

b. Click on Run from the pop-up menu.

c. In the Command Line box, type *x:setup* ("x" is the letter of your CD-ROM drive).

d. Click on the OK button, and follow the on-screen instructions.

After PipeLine is successfully installed, the PipeLine shortcut (shown below) will be displayed on your desktop.

Installing PipeLine from the Internet

PipeLine is available on the USAID | DELIVER PROJECT website at the following web address:

http://deliver.jsi.com.

PipeLine CD-Rom
If your Internet connection is slow and/or unreliable, order the PipeLine CD-ROM.

To download PipeLine—

1. Access the Internet, and enter the USAID | DELIVER PROJECT web address.
2. Locate the PipeLine download page, and follow the on-screen instructions to download PipeLine.

How to Start PipeLine

PipeLine can be started from the Windows desktop or the Windows taskbar.

Starting PipeLine from the Windows desktop

From the Windows desktop—

1. Locate and double-click on the PipeLine icon to start the application.

Starting PipeLine from the Windows taskbar

From the Windows taskbar—

1. Click on Start.
2. Click on Programs.
3. Locate and click on the PipeLine 5.0 link.

Reinstalling PipeLine

To reinstall PipeLine—

1. Place the PipeLine CD in your CD-ROM drive.

If you do not have the PipeLine CD, you can use the copy of PipeLine you downloaded from the USAID | DELIVER PROJECT website.

2. Start the install process, and follow the instructions on your screen.

Password
The Internet version of PipeLine requires a password to start the install process. That password was sent to you by email when you downloaded PipeLine. If you no longer have the email containing the PipeLine password, download PipeLine from the USAID | DELIVER PROJECT website: (http://deliver.jsi.com).

During the process, a message box is displayed instructing you to remove PipeLine from your computer.

3. Click on the Remove button to remove PipeLine from your computer.
4. Click on the Finish button when prompted.

After PipeLine has been removed—

5. Repeat the PipeLine installation procedure.

See page 27 for information on installing PipeLine.

Converting Your Existing Data Files

This version of PipeLine allows you to convert data created with previous versions of PipeLine.

From the Program Data screen—

1. Click on the File Menu Bar option, and select the Open option from the pull-down menu.

PipeLine opens a window so you can locate the data you need to convert.

Upgrading
The current program's data file is not the current version. You can allow PipeLine to upgrade the file now. If you do not, some of PipeLine's features may not work properly.

2. Locate and select the data file you need, and click on the Open button.

After you select the data file you need to convert, PipeLine displays a message similar to the one in the text box below.

3. Click on the Yes button to convert the selected data.

PipeLine opens a window, and allows you to rename the file you selected to upgrade. This safeguards the original data by saving the upgraded data under a different name.

4. Type the new name in the File Name field, and click on the Open button.

PipeLine converts the selected data, renames the file, and displays its associated program data on the Program Data screen. You can now work with the converted data file.

Original Data
Remember, your original data remains in its original directory with its original name. The converted data is a copy of the original.

Run-time installation

Table 3 - Installation Locations and Purpose

Directory/File	Purpose
PipeLine	**Parent directory**
/ANYMOH	**Directory for sample database**
/globalmoh.MDB	Sample Database
/Data	**Directory for live databases**
/Graphics	**Directory for application graphics**
/SplashNewT.avi	Splash screen movie
/PL40.ico	PipeLine taskbar icon
/Pipeline_ICON-xx.ico	PipeLine Desktop icon
/Import	**Directory for imported files**
/Summary	**Directory for PipeLine Summary**
/Roboex32.dll	Dll required for PipeLine Summary to run properly
/Proc2000.mdb	PipeLine Summary frontend
/Proc_BE.mdb	PipeLine Summary backend
/Prog2000.mdb	PipeLine Summary program list
/Summary.ico	PipeLine Summary icon
/Sumv2.cnt	PipeLine Summary help cnt file
/SUMv2.hlp	PipeLine Summary help file
/XML	**Directory for xml files**
/ECatalog_Live_Final_Generic_20100701.xml	E-Catalog file distributed with application
/SCMS Product_ARV_TEST.xml	SCMS ARV file distributed with application
/Contraceptives.xml	Contraceptives file distributed with application
/e-help.cnt	PipeLine (English) help cnt file
/e-help.HLP	PipeLine (English) help file
/E-PL-help.cnt	PipeLine help cnt file
/E-PL-help.hlp	PipeLine help file
/Pipeline_ICON-xx.ico	PipeLine icon
/Pipeline2000.MDB	PipeLine frontend file
/PLFix1.reg	Registry fix for graphs
/PLFix2.reg	Registry fix for graphs
/PLFix3.reg	Registry fix for graphs
/pmp_mpty.mdb	Empty PipeLine backend file
/posttransform.xslt	
/ProgV4.mdb	PipeLine program list
/ReadMe.txt	Readme file for installation issues and known issues
/Roboex32.dll	Dll required for PipeLine to run properly

Development environment installation

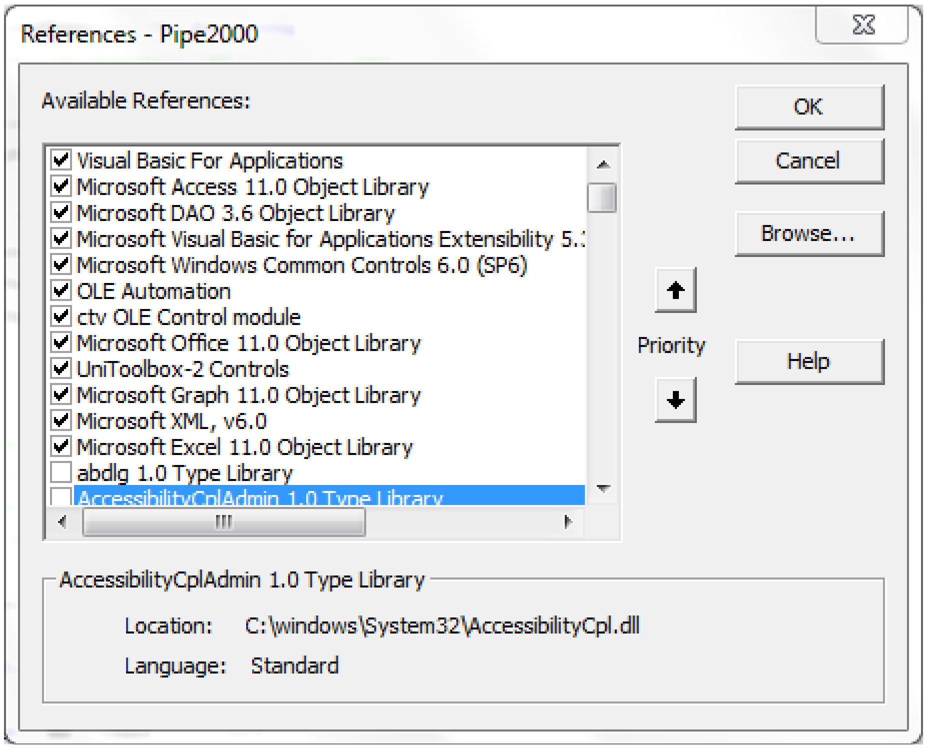
The Development Environment requires the installation of Microsoft Office 2003 (MSACCESS and Excel are required). Some other tools are suggested as well

Table 4 - Recommended Software Tools

Software/Tool	Required	Description
Microsoft Access	Y	The Main Development Tool for PipeLine
Microsoft Excel	Y	Required for the creation of the Output to Excel reports
Microsoft Visual SourceSafe (VSS)	N	The software code repository for PipeLine. Required for checking in/out the source code files. Optional (but suggested) if development is being done in a stand alone environment
Microsoft Access Plug-In: Source Code Control	N	odc_accscc.exe, this allows VSS integration into Access. Access .mdb files can be stored in VSS as individual components allowing multi-user development on a single .mdb file).
FMS Total Visual Code Tools	N	This plugin/toolbar provides the developer with the ability to quickly stub in new procedures and functions and to cleanup existing code modules with proper development standards.
Microsoft Windows Common Controls 6.0 (SP6)	Yes	This provides the Treeview control.

The Access References should look like this:

Figure 2 - PipeLine References



References - Pipe2000

Available References:

- [x] Visual Basic For Applications
- [x] Microsoft Access 11.0 Object Library
- [x] Microsoft DAO 3.6 Object Library
- [x] Microsoft Visual Basic for Applications Extensibility 5.:
- [x] Microsoft Windows Common Controls 6.0 (SP6)
- [x] OLE Automation
- [x] ctv OLE Control module
- [x] Microsoft Office 11.0 Object Library
- [x] UniToolbox-2 Controls
- [x] Microsoft Graph 11.0 Object Library
- [x] Microsoft XML, v6.0
- [x] Microsoft Excel 11.0 Object Library
- [ ] abdlg 1.0 Type Library
- [ ] AccessibilityCplAdmin 1.0 Type Library

AccessibilityCplAdmin 1.0 Type Library

Location: C:\windows\System32\AccessibilityCpl.dll

Language: Standard

Build process

The Build Process for PipeLine involves checking the code into VSS (if necessary), removing the application for VSS, compact the application, and updating the tblSysParameters. In the tblSysParameters, update the AsOf date, the Version number, and set InitialInstall to True. Please note that the application will not relink properly if InitialInstall is not set to true since this flag informs the application to look for the default data file distributed with PipeLine.

Installer

PipeLine is widely used in many countries. It is also downloaded through the web. A majority of the users install the PipeLine application on their own. The installer is first created using SageKey MSI Wizard 2003 along with Installshield 12. In order to then provide a simple, user friendly and self-installable interface, a tool called AutoRun Pro Enterprise II was used. (This tool is available at this website http://www.longtion.com.) These file create the cd image and the files are zipped using WinZip Self Extractor to create a single downloadable file used for the web distribution.

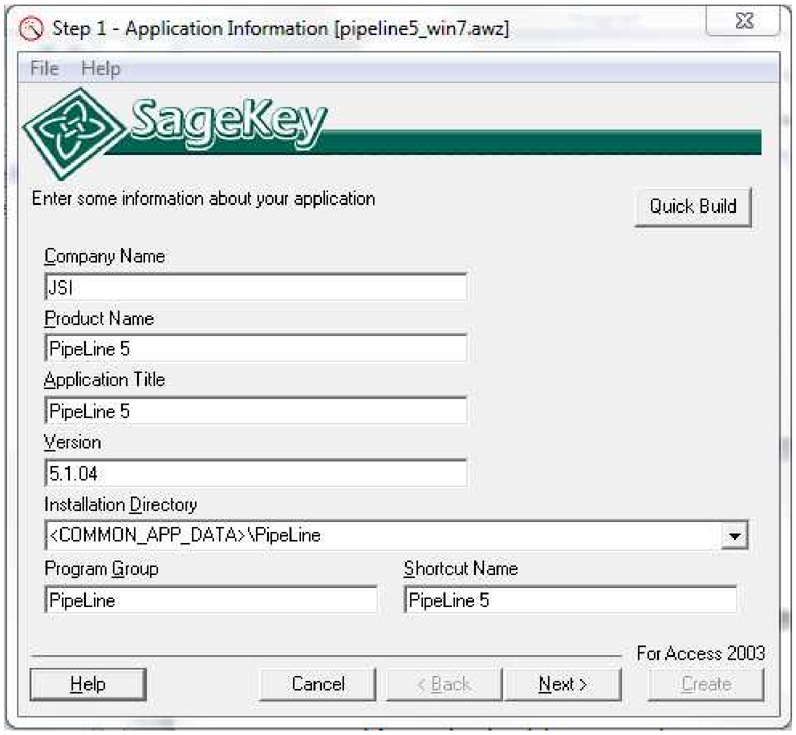
Mechanics for Creating the CD Installer

Create files with SageKey

Enter Application Information

Once the build is complete, open the file PipeLine5_win7.awz.

Figure 3 - Application Information


Step 1 - Application Information [pipeline5_win7.awz]

Enter some information about your application

Company Name: JSI

Product Name: PipeLine 5

Application Title: PipeLine 5

Version: 5.1.04

Installation Directory: <COMMON_APP_DATA>\PipeLine

Program Group: PipeLine

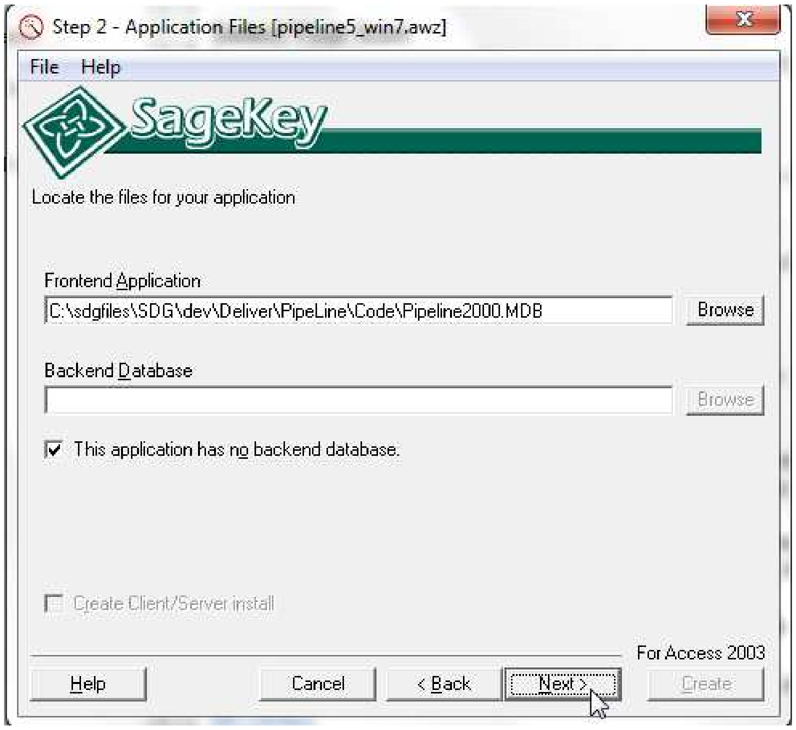
Shortcut Name: PipeLine 5

For Access 2003

Select Application Files

Update the basic information about PipeLine here. For more information on each field, right click the corresponding label and select 'What's this?" Click Next to go to next screen.

Figure 4 - Application Files


Step 2 - Application Files [pipeline5_win7.awz]

Locate the files for your application

Frontend Application: C:\sdgfiles\SDG\dev\Deliver\PipeLine\Code\Pipeline2000.MDB

Backend Database:

[✓] This application has no backend database.

[ ] Create Client/Server install

For Access 2003

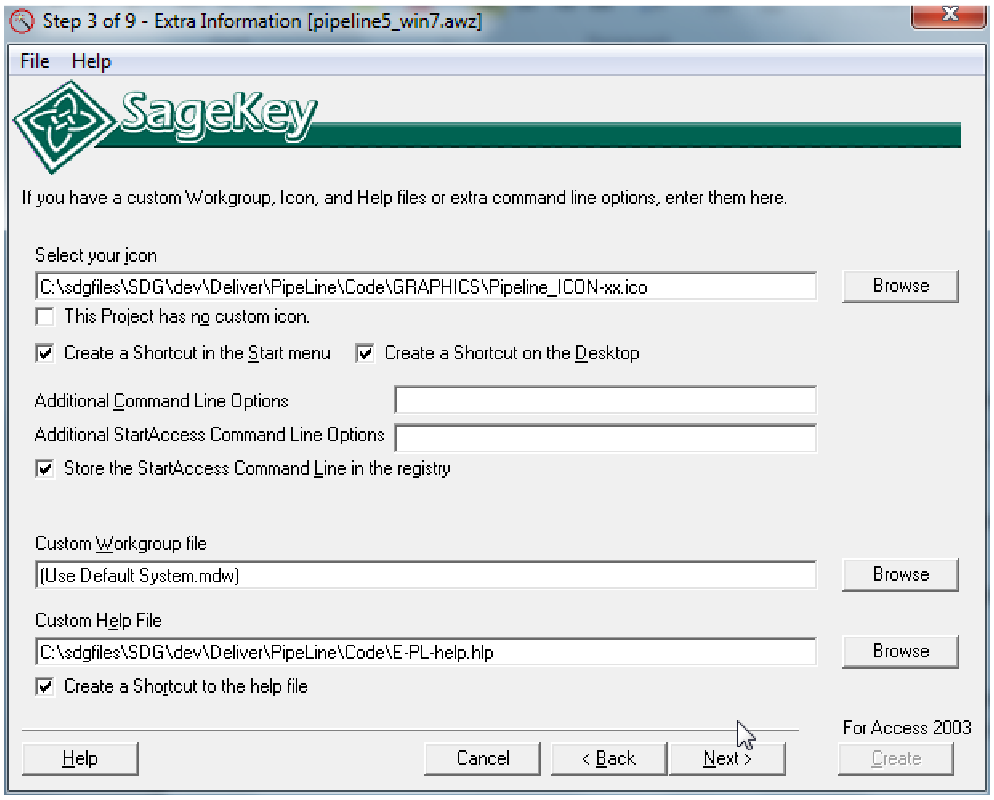
Select your database project files here and click Next to go to next screen.

Front-end Application
Click 'Browse' to locate the PipeLine2000.mdb file. The files must exist in the place specified or you can not proceed to the next step.

Back-end Database
Since we connect to the backend dynamically, check the box next to 'This application has no back-end database.' The text field will go blank and the browse button will no longer be available.

Enter Extra Information

Figure 5 - Extra Information



If you have a custom Workgroup, Icon, and Help files or extra command line options, enter them here.

Select your icon: C:\sdgfiles\SDG\dev\Deliver\PipeLine\Code\GRAPHICS\Pipeline_ICON-xx.ico

☐ This Project has no custom icon.

☑ Create a Shortcut in the Start menu ☑ Create a Shortcut on the Desktop

Additional Command Line Options:

Additional StartAccess Command Line Options:

☑ Store the StartAccess Command Line in the registry

Custom Workgroup file: (Use Default System.mdw)

Custom Help File: C:\sdgfiles\SDG\dev\Deliver\PipeLine\Code\E-PL-help.hlp

☑ Create a Shortcut to the help file

For Access 2003

Select Your Icon
Browse to the PipeLine icon found in the graphics folder. The file must exist to proceed to the next step.

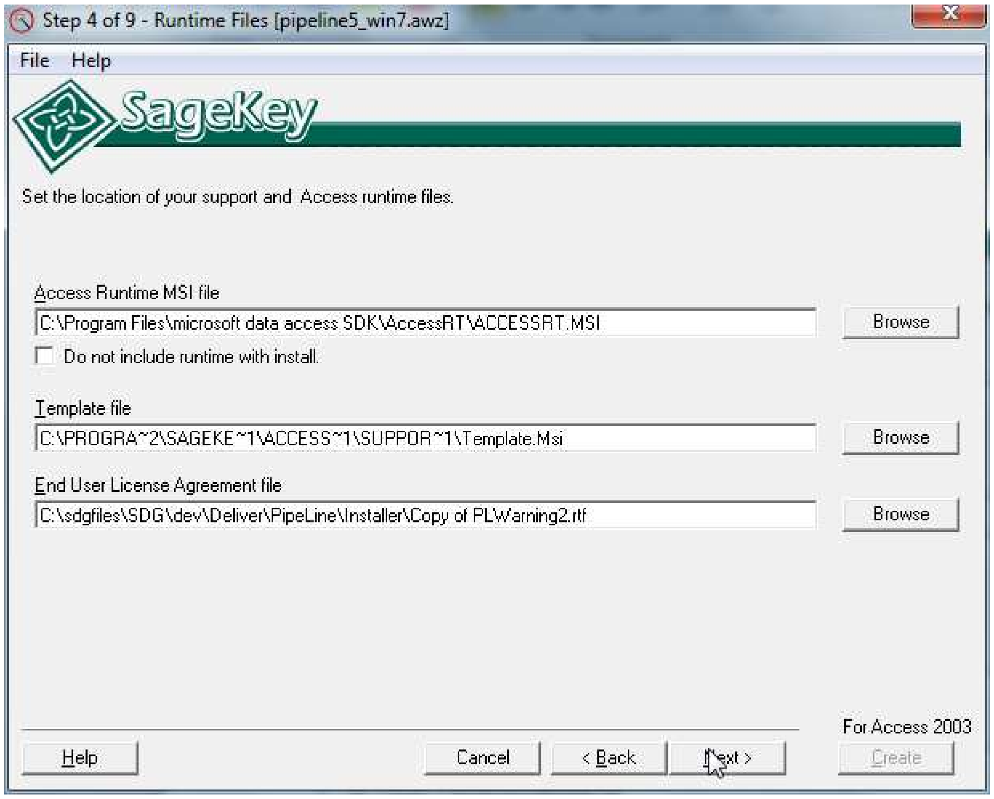
Select Shortcuts
Choose the Desktop and the Start Menu shortcuts.

Store the StartAccess Command Line in the registry & System.mdw
Choose to stre the startaccess command line in the registry. And use the default system.mdw

Custom Help File
Browse to the PipeLine help file. If you want a shortcut to the help file on the Start Menu check the box.

Select Runtime Files

Figure 6 - Runtime Files



Step 4 of 9 - Runtime Files [pipeline5_win7.awz]

Set the location of your support and Access runtime files.

Access Runtime MSI file: C:\Program Files\microsoft data access SDK\AccessRT\ACCESSRT.MSI

☐ Do not include runtime with install.

Template file: C:\PROGRA~2\SAGEKE~1\ACCESS~1\SUPPOR~1\Template.Msi

End User License Agreement file: C:\sdgfiles\SDG\dev\Deliver\PipeLine\Installer\Copy of PLWarning2.rtf

For Access 2003

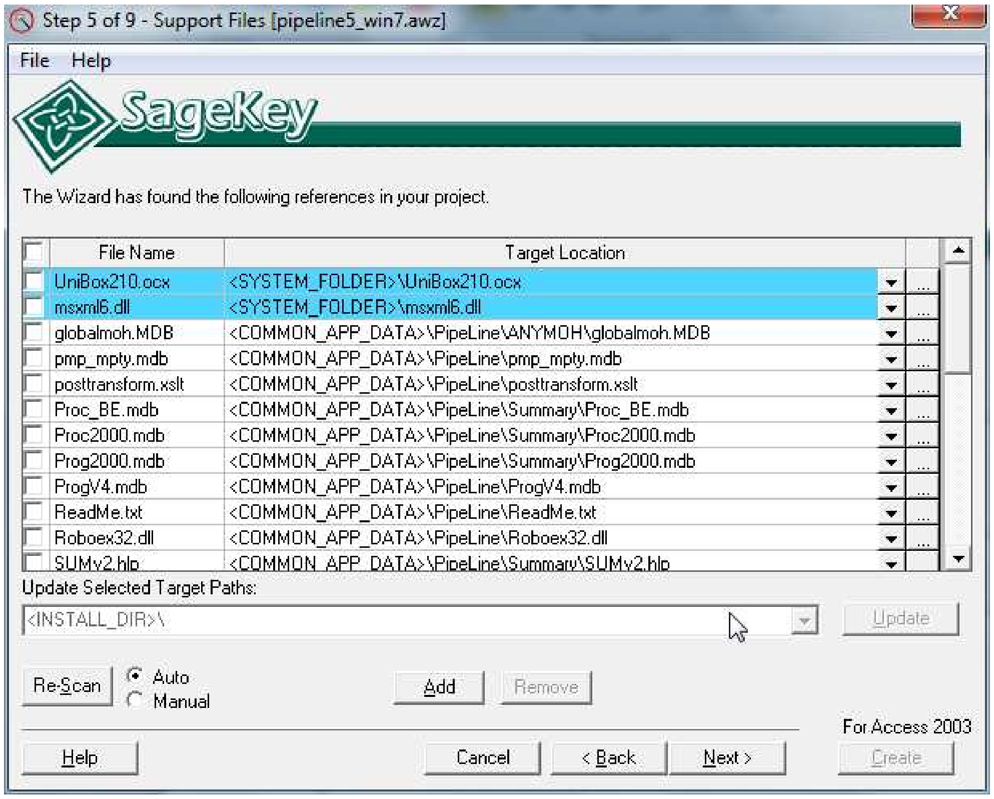
Access Runtime Files:
Browse to location of the 'Accessrt.msi'.

Template File:
Browse to the directory that contains the wizard support files and locate the file "Template.msi". (The installation of the Wizard will set this field to the default directory \Program Files\SageKey Software\Access 2003 MSI Wizard\Support Files\Template.msi)

End User License Agreement file:
Browse ot the PipeLine Warning file in the installer directory.

Select Other Files

Figure 7 - Support Files


Step 5 of 9 - Support Files [pipeline5_win7.awz]

The Wizard has found the following references in your project.

File Name	Target Location
UniBox210.ocx	<SYSTEM_FOLDER>\UniBox210.ocx
msxml6.dll	<SYSTEM_FOLDER>\msxml6.dll
globalmoh.MDB	<COMMON_APP_DATA>\PipeLine\ANYMOH\globalmoh.MDB
pmp_mpty.mdb	<COMMON_APP_DATA>\PipeLine\pmp_mpty.mdb
posttransform.xslt	<COMMON_APP_DATA>\PipeLine\posttransform.xslt
Proc_BE.mdb	<COMMON_APP_DATA>\PipeLine\Summary\Proc_BE.mdb
Proc2000.mdb	<COMMON_APP_DATA>\PipeLine\Summary\Proc2000.mdb
Prog2000.mdb	<COMMON_APP_DATA>\PipeLine\Summary\Prog2000.mdb
ProgV4.mdb	<COMMON_APP_DATA>\PipeLine\ProgV4.mdb
ReadMe.txt	<COMMON_APP_DATA>\PipeLine\ReadMe.txt
Roboex32.dll	<COMMON_APP_DATA>\PipeLine\Roboex32.dll
SUMv2.hlp	<COMMON_APP_DATA>\PipeLine\Summary\SUMv2.hlp

Update Selected Target Paths: <INSTALL_DIR>\

(● Auto ○ Manual)

For Access 2003

When you click 'Next' in Step 4 (Runtime Files), the Wizard opens the file you specified as your Front-end in Step 2 (Application Files) and scans it for references. If it finds any, they are added here. You can remove any erroneous ones, or, if you make changes at a later time, click the 'Re-Scan' button to refresh the list.

The Wizard will try to determine the target path for each of these, but you can change the installation path by editing the Target Path combo field. You can edit this and change it, but do not change or add new variables. (i.e.: Don't change or remove <PROGRAM_FILES> or any other words enclosed in less than (<) or greater than (>) brackets.) This will cause the Installation build to fail.

Any additional files your application needs that are not included as references are added here as well. Clicking the 'Add' button will bring up the Add File dialog.

You can select multiple files to add in this dialog. Once you have selected the file or files you want to add to the install, click the Open button to add these files.

By Default, each file you selected will have a target directory of <INSTALL_DIR>\filename. You can change the target directory for multiple files by checking the checkbox beside each file to change and entering a new target path in the "Update Selected Target Path" combo box. Press the Update button to update the target path of the selected files.

File Details

At the right hand side of each file row, there is a button which will display the File Details dialog.

This dialog allows you to specify the Source Path of the file in the case that you have moved the source files for you project. Also, you can check the "Show Advanced Options" checkbox to view the advanced options for the specific file.

Please Note: Changing any of the advanced fields is only recommended for users who are very familiar with msi.

Attributes: This is the Attributes column for the Component this file will belong to. For more information, see:

http://msdn2.microsoft.com/en-us/library/aa368007.aspx

Install Condition: This is the Condition column for the Component this file will belong to. For more information, see:

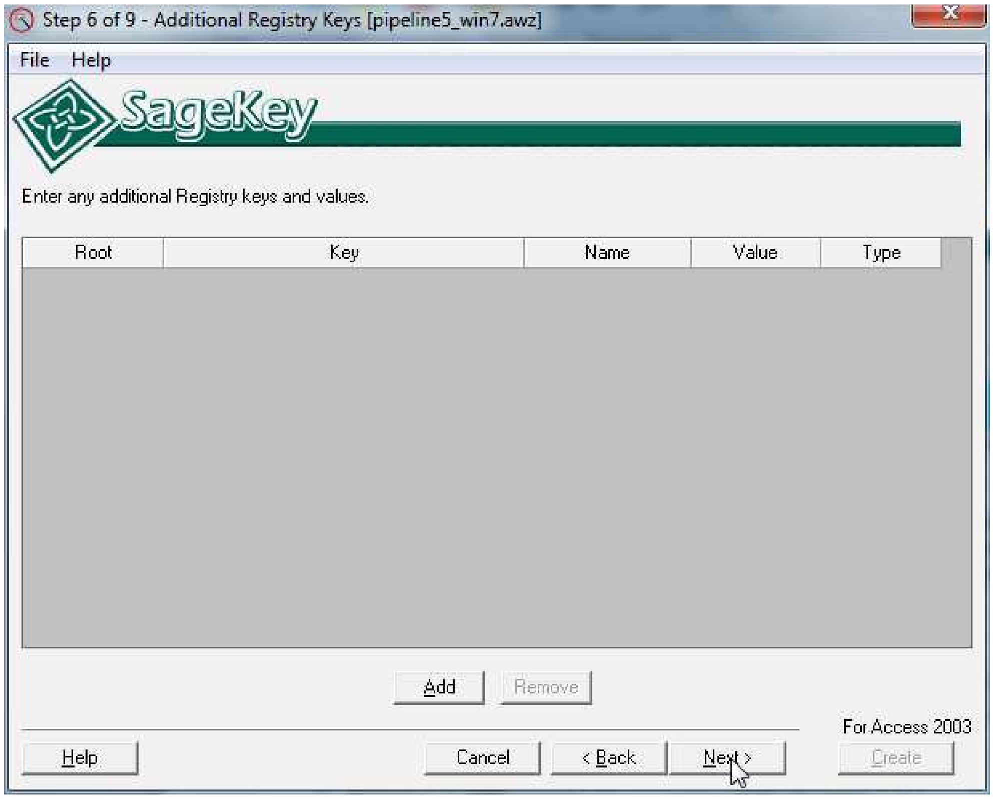
http://msdn2.microsoft.com/en-us/library/aa368007.aspx

Re-Scan

By Default, each time you go through the wizard for an existing project, the front-end database will be scanned for references. This can be changed by selecting the "Manual" radio button beside the Re-Scan button. If Manual is chosen, then the front-end database will not be automatically scanned for references.

Enter Registry Keys

Figure 8 - Additional Registry Keys

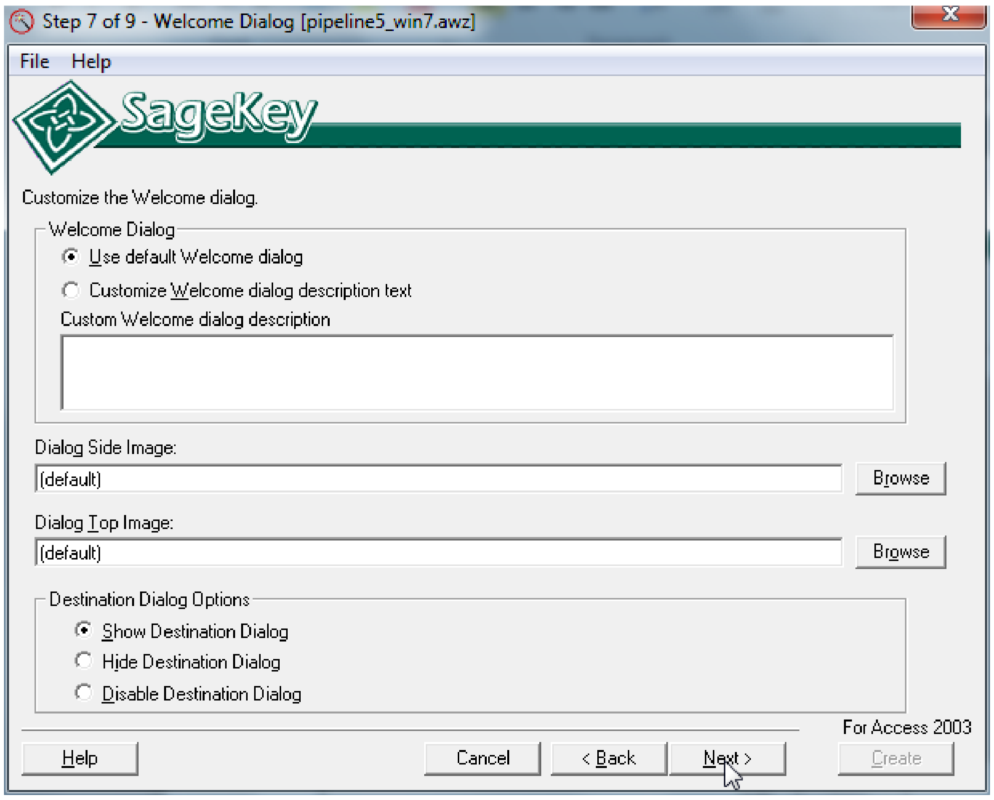


Enter any additional Registry keys and values.

Since PipeLine does not need any special registry keys, click next to continue.

Design Welcome Dialog

Figure 9 - Welcome Dialog


Step 7 of 9 - Welcome Dialog [pipeline5_win7.awz]

Customize the Welcome dialog.

Welcome Dialog
- (●) Use default Welcome dialog
- ( ) Customize Welcome dialog description text

Custom Welcome dialog description:

Dialog Side Image: (default)

Dialog Top Image: (default)

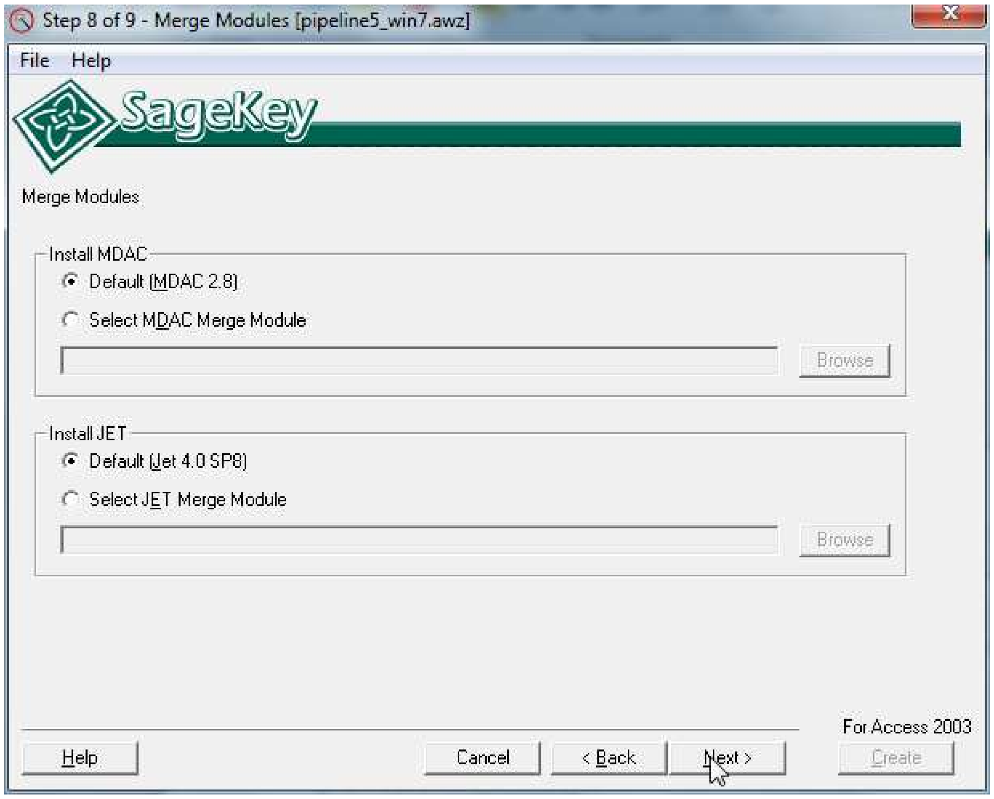
Destination Dialog Options
- (●) Show Destination Dialog
- ( ) Hide Destination Dialog
- ( ) Disable Destination Dialog

For Access 2003

Choose to use the default welcome dialog and show destination dialog and click Next.

Select Merge Modules

Figure 10 - Merge Modules



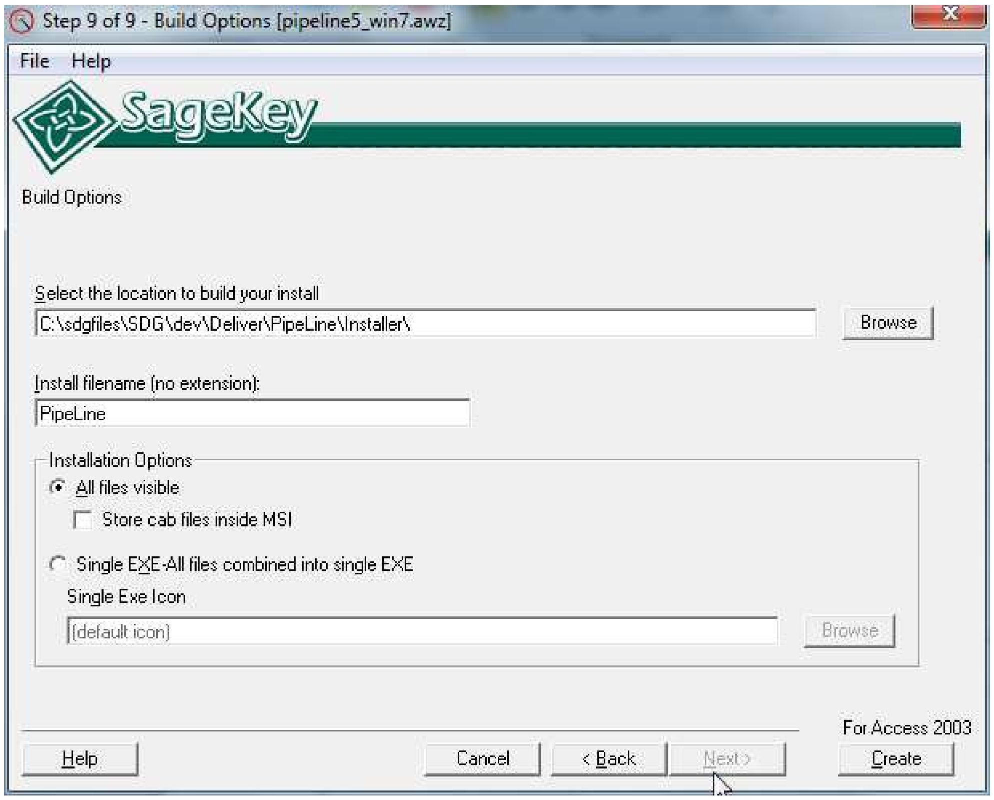
Install MDAC
(● Default (MDAC 2.8)
○ Select MDAC Merge Module

Install JET
● Default (Jet 4.0 SP8)
○ Select JET Merge Module

Choose to use default MDAC and Jet and click next.

Enter Build Options

Figure 11 - Build Options



Select the location to build your install: C:\sdgfiles\SDG\dev\Deliver\PipeLine\Installer\

Install filename (no extension): PipeLine

Installation Options
● All files visible
☐ Store cab files inside MSI
○ Single EXE-All files combined into single EXE
Single Exe Icon: (default icon)

Build Location:
Choose where the Wizard will build the Microsoft Windows Installer (.msi) file.

Install File Name
Type the name you wish to use for the main install file Do not type an extension for this file, it will be added by the Wizard.

Installation Options
All Files Visible
Select all files visble to have all of the files visible in the installation directory. You will need to include:

- -[InstallFilename].exe
- -Install.ini
- -the files subfolder

Create
Click the Create Button and the Wizard will verify if all the properties look correct, if so, it then builds the Install.

Create Interface with AutoRun Pro

Download and Install the Application

Download the application from the link below:

http://www.longtion.com/autorunenterpriseii/autorunpro.htm

Install the application. The installation steps are simple, like any other windows application, wizard driven, and takes under 2 minutes.

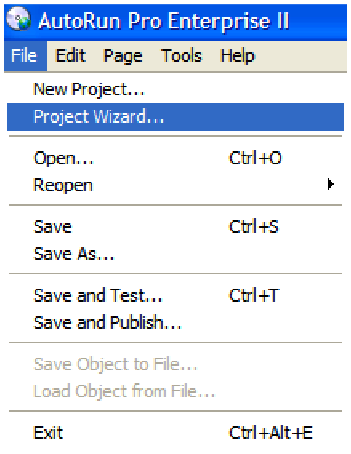
Once the application is installed, start the application and start building a project.

Create a New Project

Start the Application
Find and click on the AutoRun Pro icon on your desktop

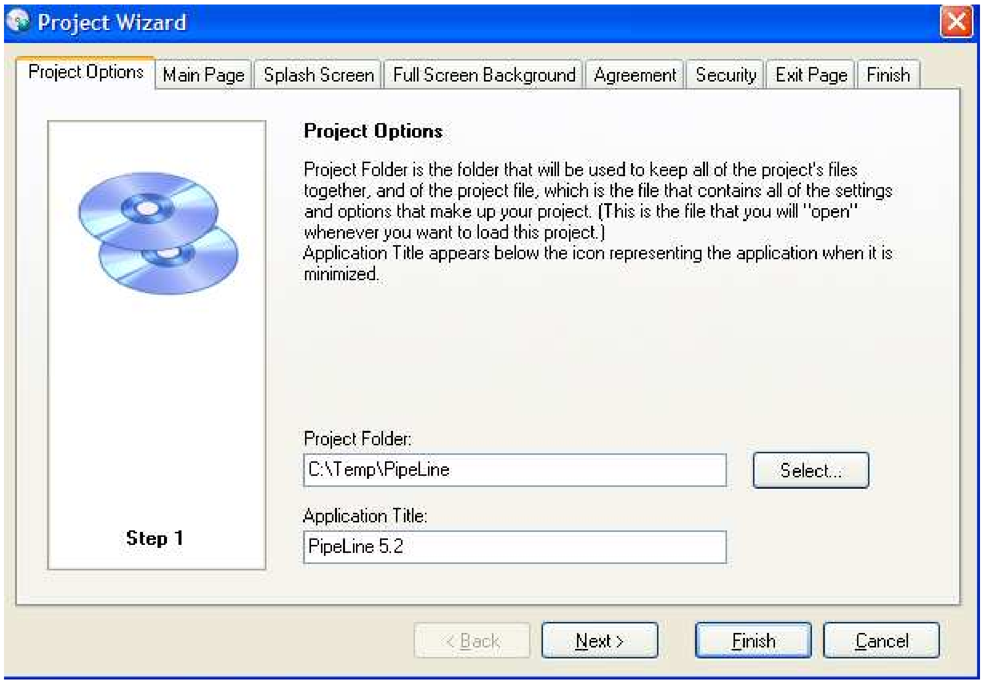
Make New Project
Create a new project to build the PipeLine CD auto-run program. See following illustrations for the steps.

Figure 12 - File Dropdown Menu



Select File > Project Wizard. The Project Wizard starts.

Figure 13 - Project Options



Project Options

Project Folder is the folder that will be used to keep all of the project's files together, and of the project file, which is the file that contains all of the settings and options that make up your project. (This is the file that you will "open" whenever you want to load this project.)
Application Title appears below the icon representing the application when it is minimized.

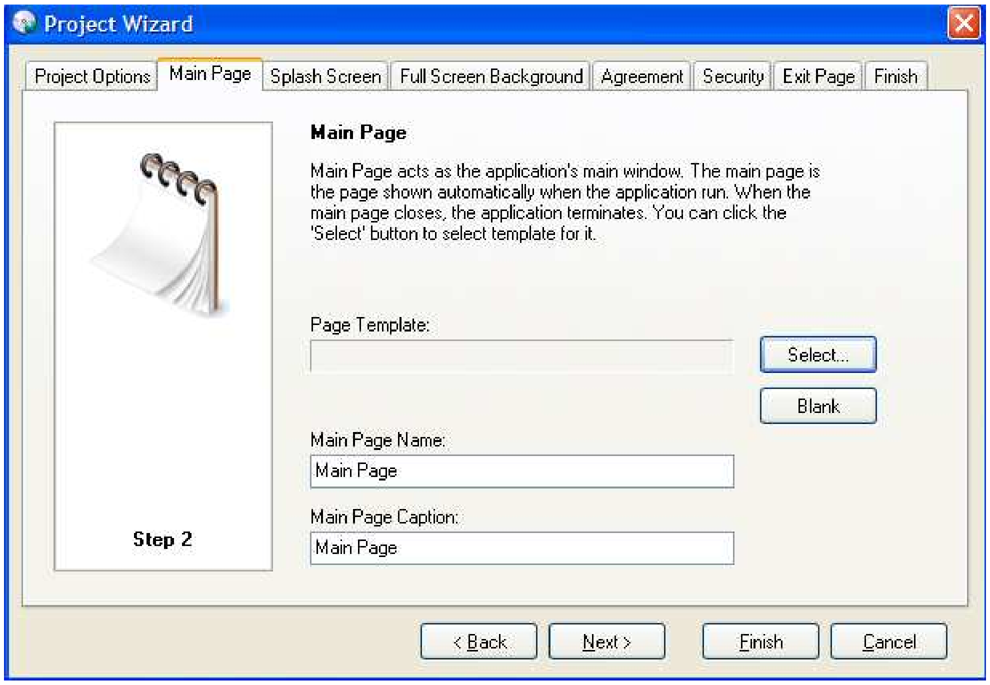
Project Folder: C:\Temp\PipeLine

Application Title: PipeLine 5.2

Select the folder on which to save the project files. Give a name to the project.

CD installer can display installation screens through multiple menu/pages. In next screen of the Project Wizard select a suitable template to base the look and feel of the installation screen.

Figure 14 – Main Page



Main Page

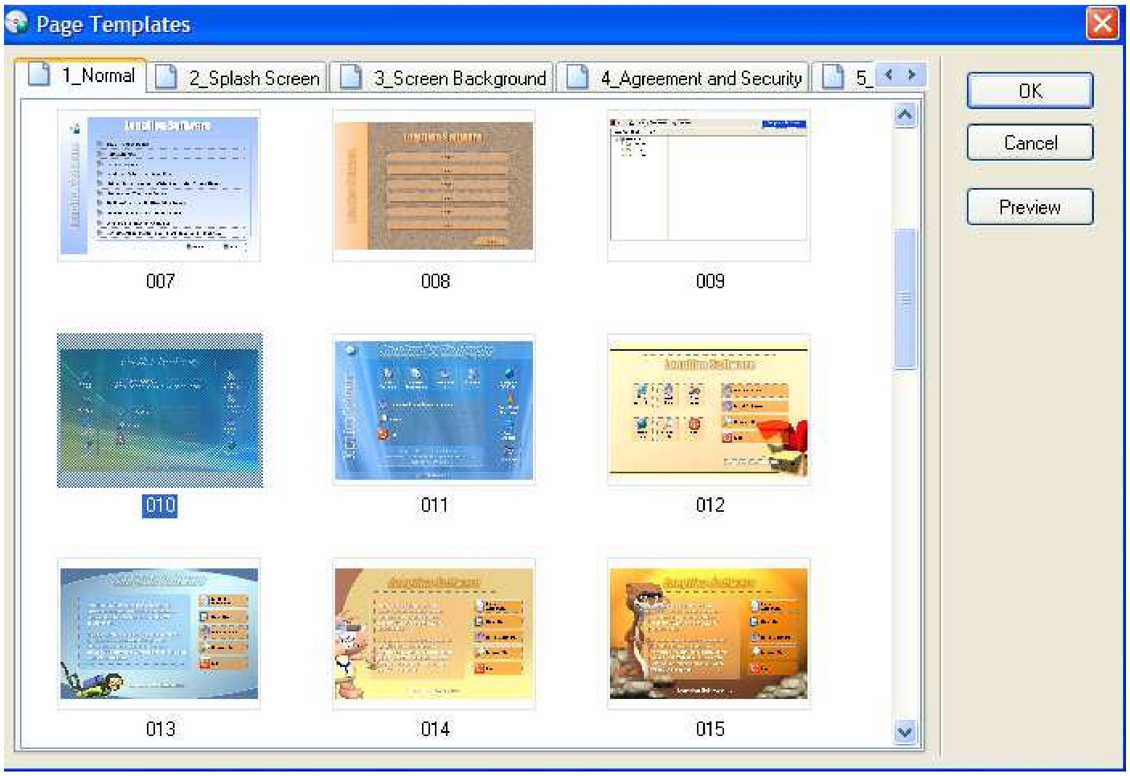
Main Page acts as the application's main window. The main page is the page shown automatically when the application run. When the main page closes, the application terminates. You can click the 'Select' button to select template for it.

Page Template:

Main Page Name: Main Page

Main Page Caption: Main Page

Figure 15 – Page Templates



Template number 010 was used as the base template for PipeLine.

Next couple of screens of the wizard asks about whether to use splash screen, what kind of background graphics to use, whether to ask the user to agree to a user agreement legal statement, a security page to enter password to activate the product, an exit page to display to the user. None of these were used for the PipeLine project. Click finish to complete the project creation steps.

Customize the template

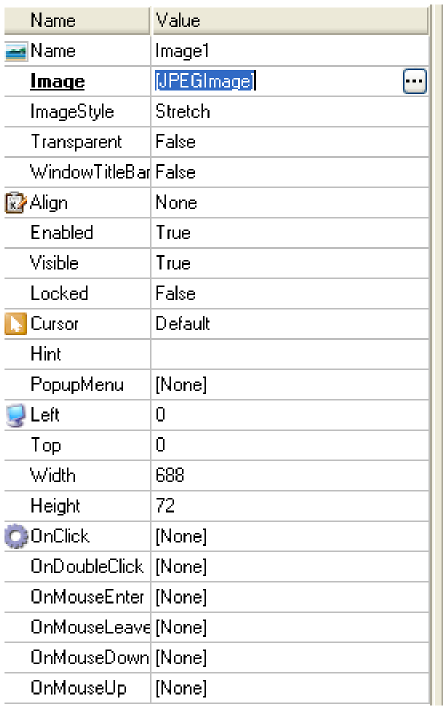
Customize the template number 010 that was used as the base. The process of customizing is basically:

- **Select a screen background image and color**. No background image was use. Color was set to white.
- **Set screen size**. Screen size was kept to default 697 X 480
- **Apply logo**. The USAID | DELIVER PROJECT logo was used. See screenshot below.
- **Create required pages**. After finalizing the page 1, a second page was created through Page > Duplicate Page menu option. The first page is meant to display program installations, links to website for further resources and a link to go to next screen. The second page displays the PipeLine documentation related menu choices.
- **Create menu options**. The template number 010 comes with default menu option choices. Those menu options were customized to build PipeLine related menu choices. The associated steps are illustrated below.

Apply logo

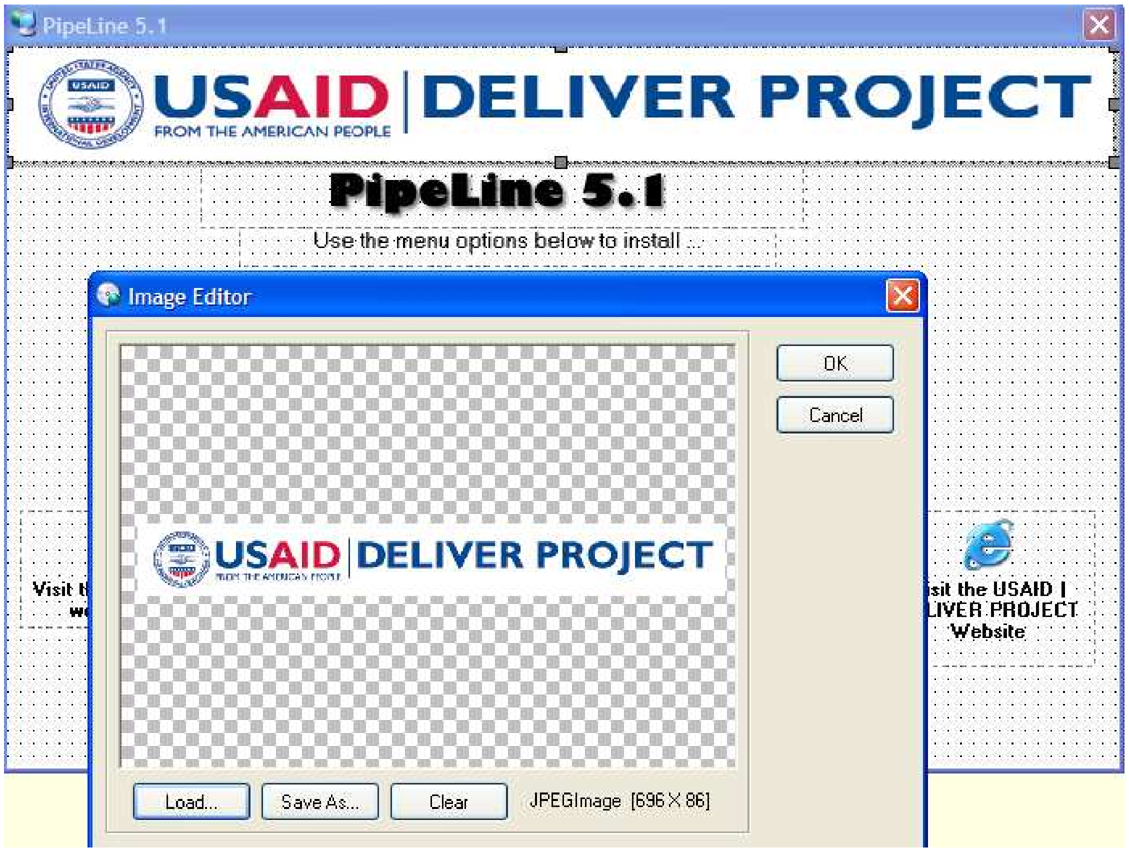
On the page 1 properties window, selecting the image item and select USAID logo. See illustration below:

Figure 16 - Properties Window



Name	Value
Name	Image1
Image	(JPEGImage)
ImageStyle	Stretch
Transparent	False
WindowTitleBar	False
Align	None
Enabled	True
Visible	True
Locked	False
Cursor	Default
Hint	
PopupMenu	[None]
Left	0
Top	0
Width	688
Height	72
OnClick	[None]
OnDoubleClick	[None]
OnMouseEnter	[None]
OnMouseLeave	[None]
OnMouseDown	[None]
OnMouseUp	[None]

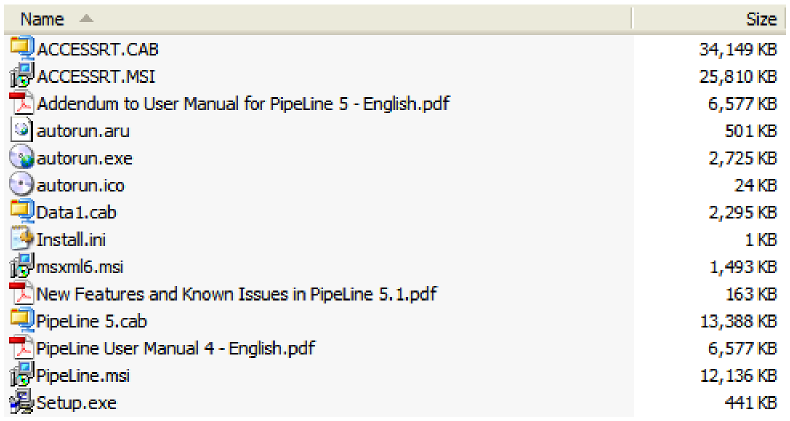
Figure 17 - Image Editor



Place PipeLine media files in a folder

Create a folder to place all files related to PipeLine program. The AutoRun program allows creating folder and sub-folder structure for complex projects. However for PipeLine project, all files were kept on a single folder, as illustrated below.

Figure 18 - PipeLine Files



Name	Size
ACCESSRT.CAB	34,149 KB
ACCESSRT.MSI	25,810 KB
Addendum to User Manual for PipeLine 5 - English.pdf	6,577 KB
autorun.aru	501 KB
autorun.exe	2,725 KB
autorun.ico	24 KB
Data1.cab	2,295 KB
Install.ini	1 KB
msxml6.msi	1,493 KB
New Features and Known Issues in PipeLine 5.1.pdf	163 KB
PipeLine 5.cab	13,388 KB
PipeLine User Manual 4 - English.pdf	6,577 KB
PipeLine.msi	12,136 KB
Setup.exe	441 KB

Create menu choices

The template 010 comes with various menu choices. The following steps illustrate the process of building the PipeLine menu choices. Instead of repeating the steps for identical items, only the unique items were illustrated below.

Create menu options that calls the PipeLine installer to run
The menu options are defined through the properties window. The visual layout and the selected properties to define the PipeLine installation main menu option are illustrated below.

Figure 19 - Visual Layout

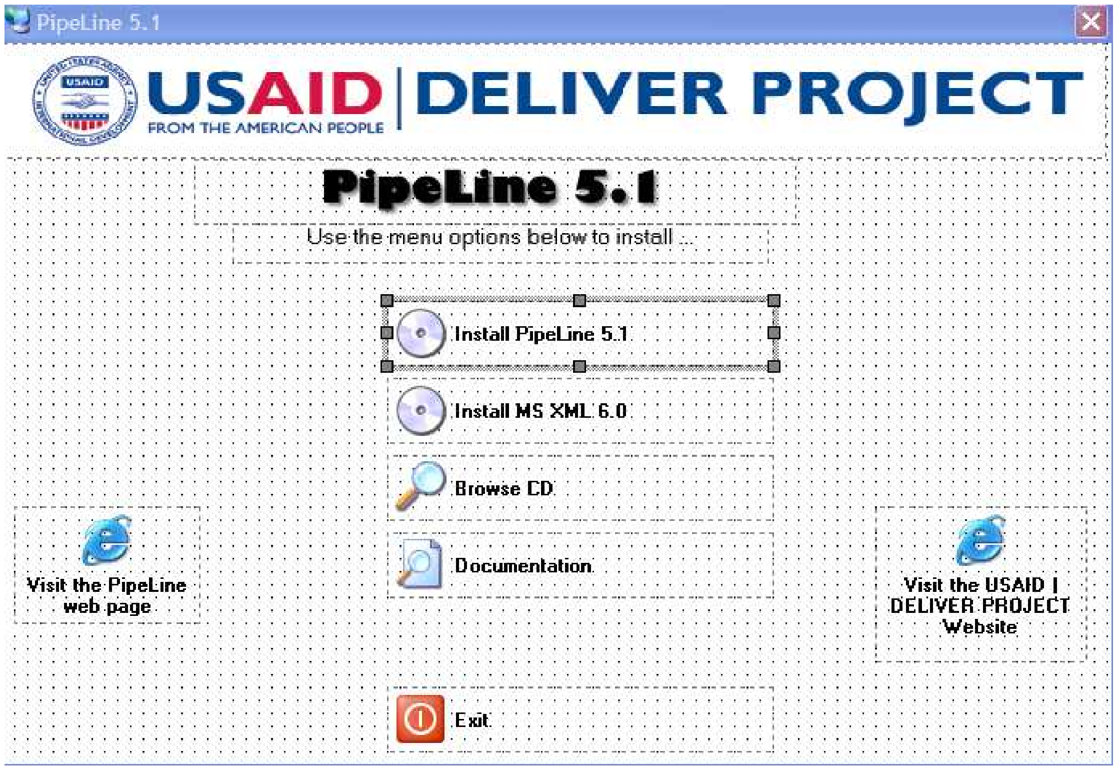
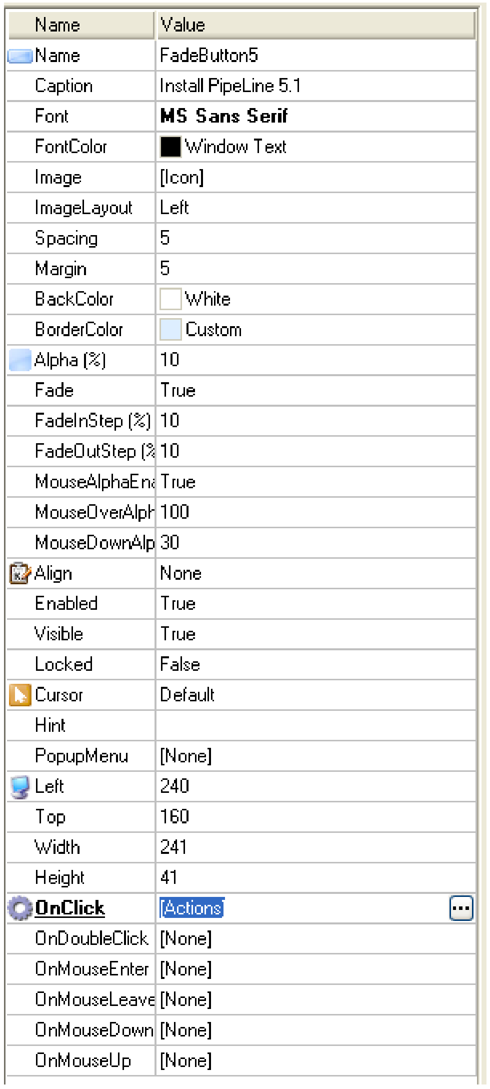

PipeLine 5.1
USAID FROM THE AMERICAN PEOPLE | DELIVER PROJECT
PipeLine 5.1
Use the menu options below to install ...
Install PipeLine 5.1
Install MS XML 6.0
Browse CD
Documentation
Visit the PipeLine web page
Visit the USAID | DELIVER PROJECT Website
Exit

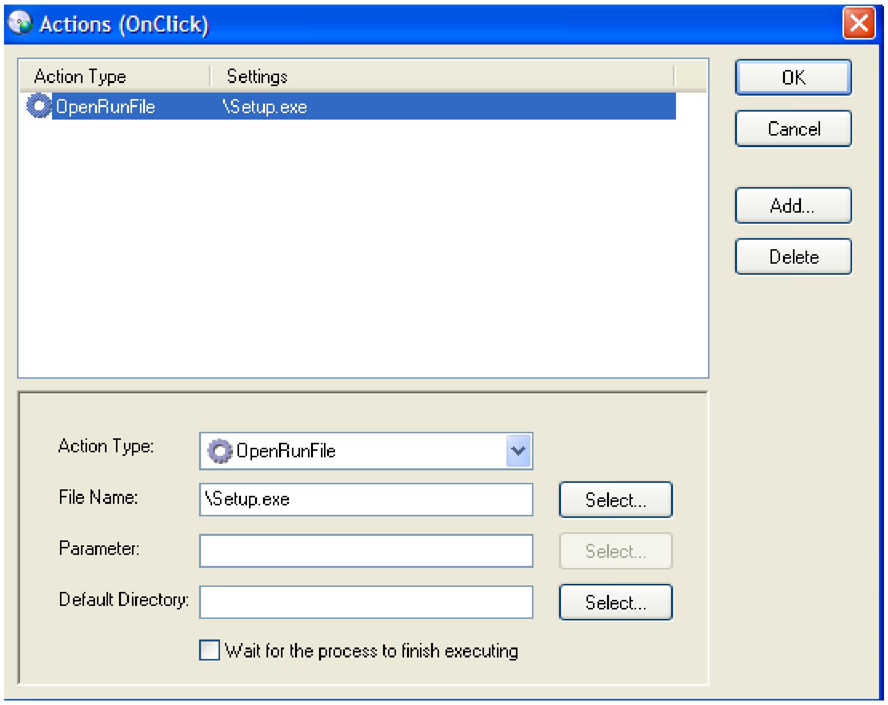
Figure 20 - Main Menu Properties



Name	Value
Name	FadeButton5
Caption	Install PipeLine 5.1
Font	**MS Sans Serif**
FontColor	■ Window Text
Image	[Icon]
ImageLayout	Left
Spacing	5
Margin	5
BackColor	□ White
BorderColor	□ Custom
Alpha (%)	10
Fade	True
FadeInStep (%)	10
FadeOutStep (%	10
MouseAlphaEn	True
MouseOverAlph	100
MouseDownAlp	30
Align	None
Enabled	True
Visible	True
Locked	False
Cursor	Default
Hint	
PopupMenu	[None]
Left	240
Top	160
Width	241
Height	41
OnClick	[Actions]
OnDoubleClick	[None]
OnMouseEnter	[None]
OnMouseLeave	[None]
OnMouseDown	[None]
OnMouseUp	[None]

Through the properties window above, text, label, icon, formatting and onclick events were defined.

Figure 21 - OnClick Actions


Actions (OnClick)

Action Type	Settings
OpenRunFile	\Setup.exe

Action Type: OpenRunFile

File Name: \Setup.exe

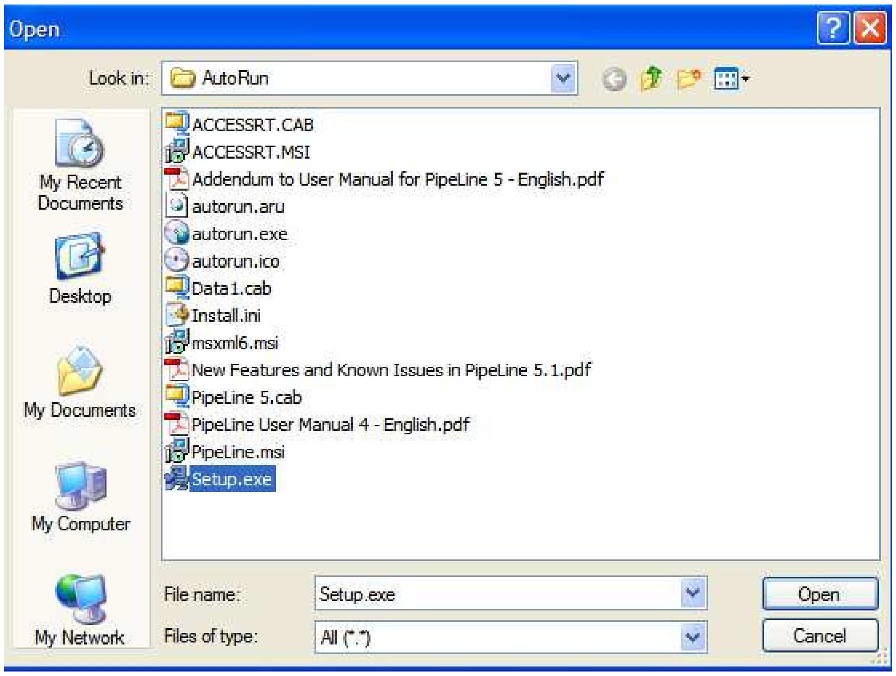
Parameter:

Default Directory:

☐ Wait for the process to finish executing

The onClick event executes the Setup.exe file placed at the root folder. Using the select button, select the Setup.exe file and click Open.

Figure 22 - File Selection Window


Open

Look in: AutoRun

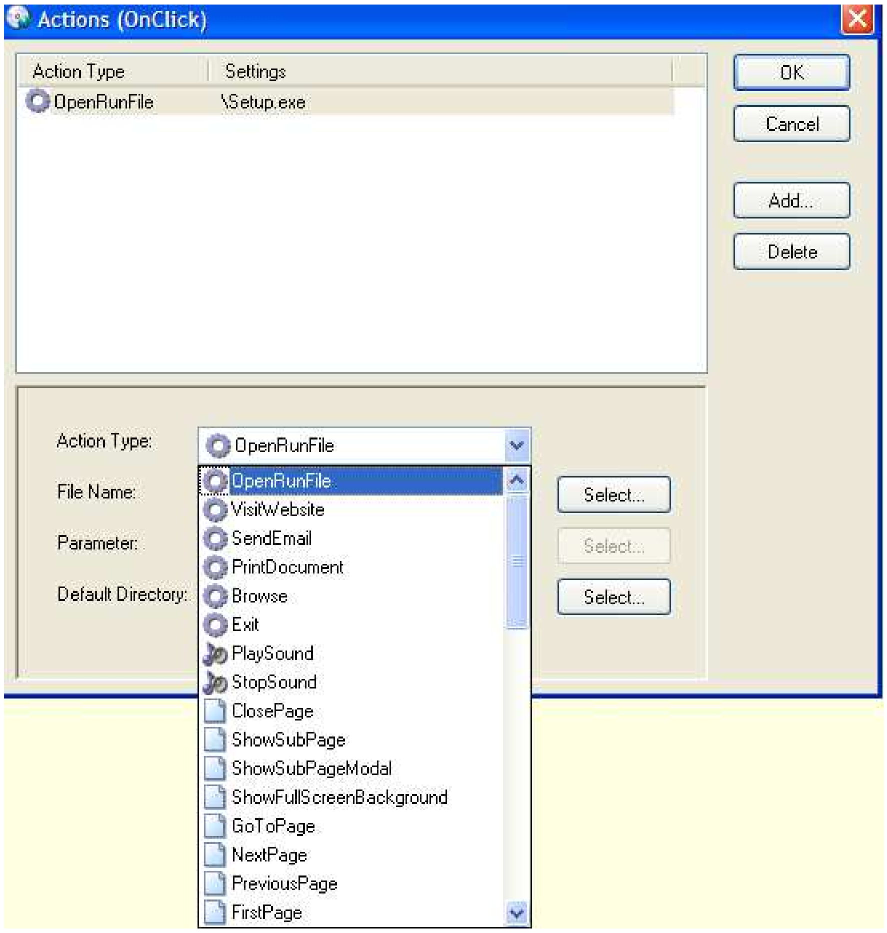
ACCESSRT.CAB
ACCESSRT.MSI
Addendum to User Manual for PipeLine 5 - English.pdf
autorun.aru
autorun.exe
autorun.ico
Data1.cab
Install.ini
msxml6.msi
New Features and Known Issues in PipeLine 5.1.pdf
PipeLine 5.cab
PipeLine User Manual 4 - English.pdf
PipeLine.msi
Setup.exe

File name: Setup.exe

Files of type: All (*.*)

Other available OnClick events are illustrated below

Figure 23 – Available OnClick Events



Actions (OnClick)

Action Type	Settings
OpenRunFile	\Setup.exe

Action Type: OpenRunFile

File Name:

Parameter:

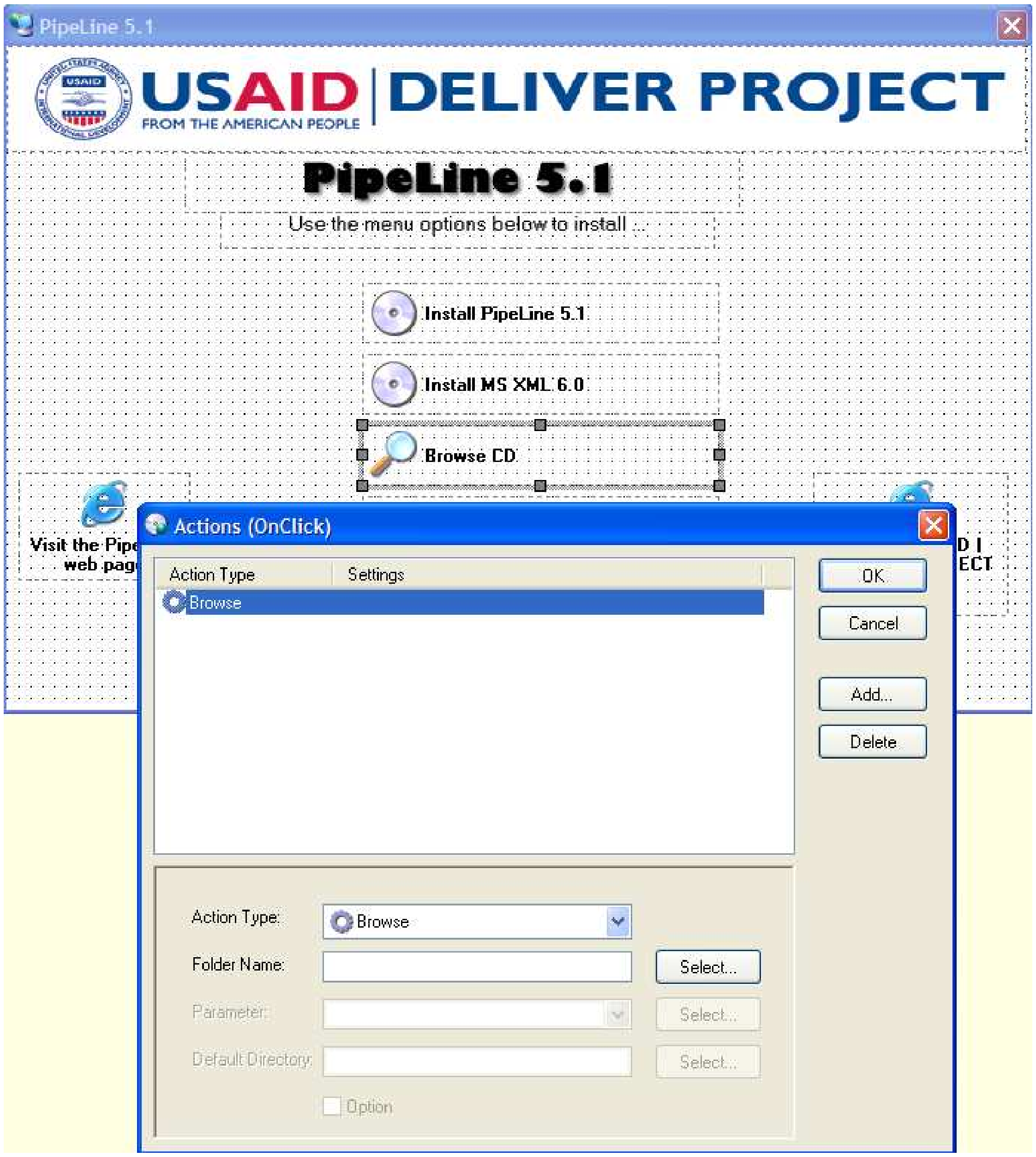
Default Directory:

OpenRunFile
VisitWebsite
SendEmail
PrintDocument
Browse
Exit
PlaySound
StopSound
ClosePage
ShowSubPage
ShowSubPageModal
ShowFullScreenBackground
GoToPage
NextPage
PreviousPage
FirstPage

Create CD browse menu option

Using Onclick > Browse event, the CD browse option was defined. See illustration below.

Figure 24 – Browse CD OnClick Action



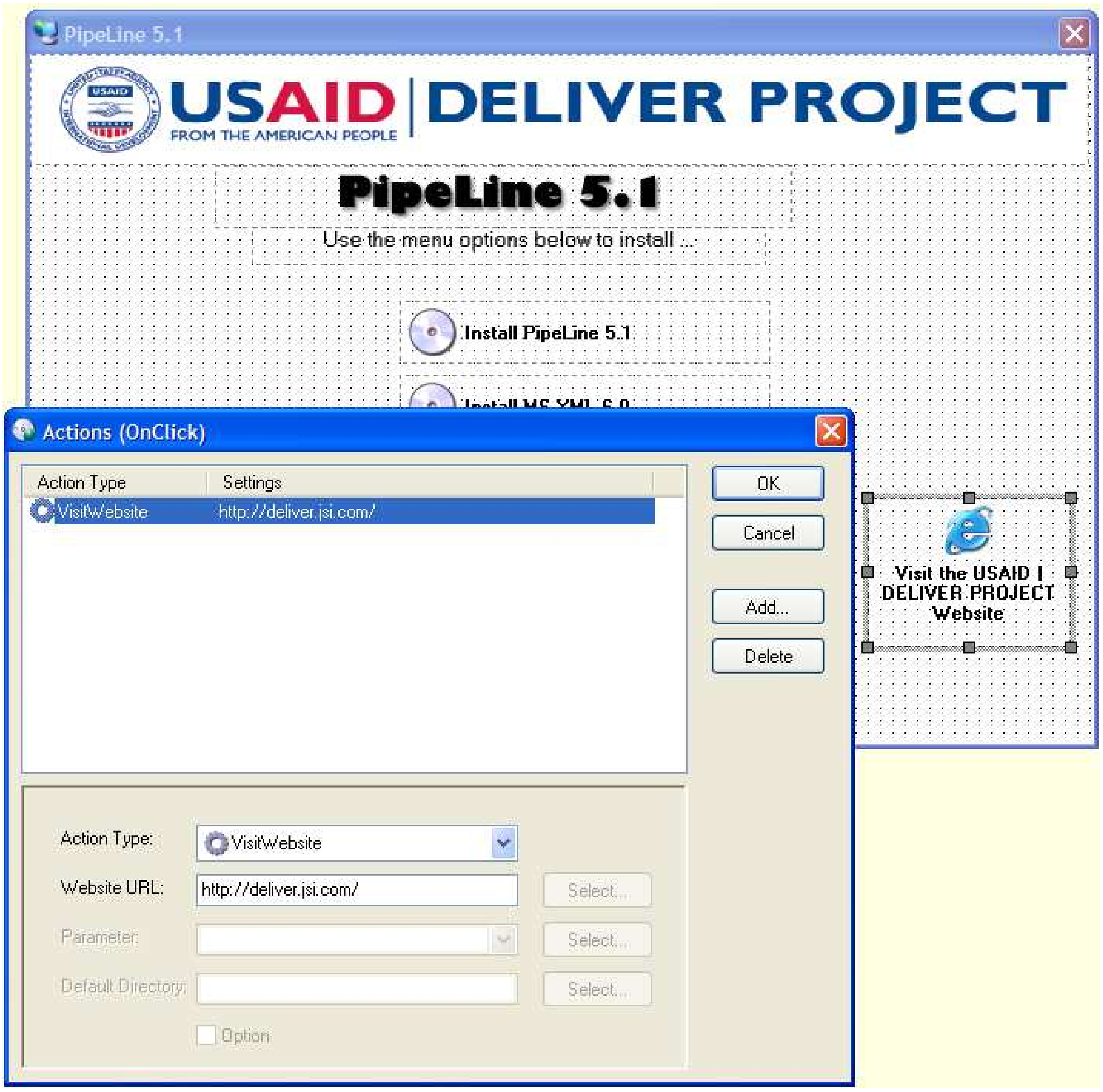
Action Type	Settings
Browse	

Action Type: Browse
Folder Name:
Parameter:
Default Directory:
Option

Create link to external website menu option

Using Onclick > VisitWebsite event, the navigation to the USAID | DELIVER PROJECT website and Visit PipeLine website links were created. This click will go to respective website, using the users' default browser.

Figure 25 - USAID|DELIVER PROJECT Website OnClick Action



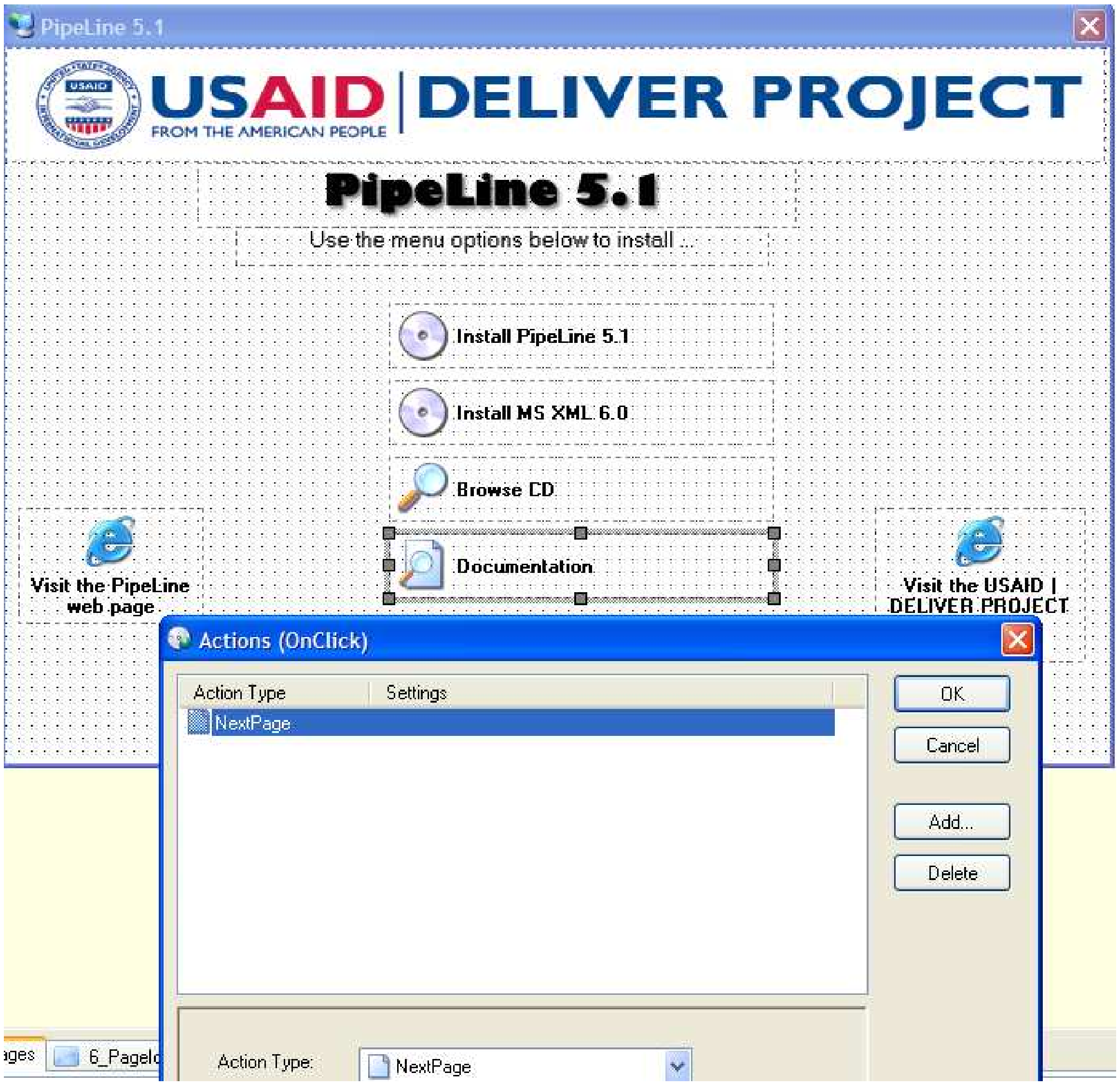
Action Type	Settings
VisitWebsite	http://deliver.jsi.com/

Action Type: VisitWebsite
Website URL: http://deliver.jsi.com/
Parameter:
Default Directory:

Create link to next page menu option

Using Onclick > NextPage event, the navigation to next page was defined. See illustration below.

Figure 26 - Documentation OnClick Action



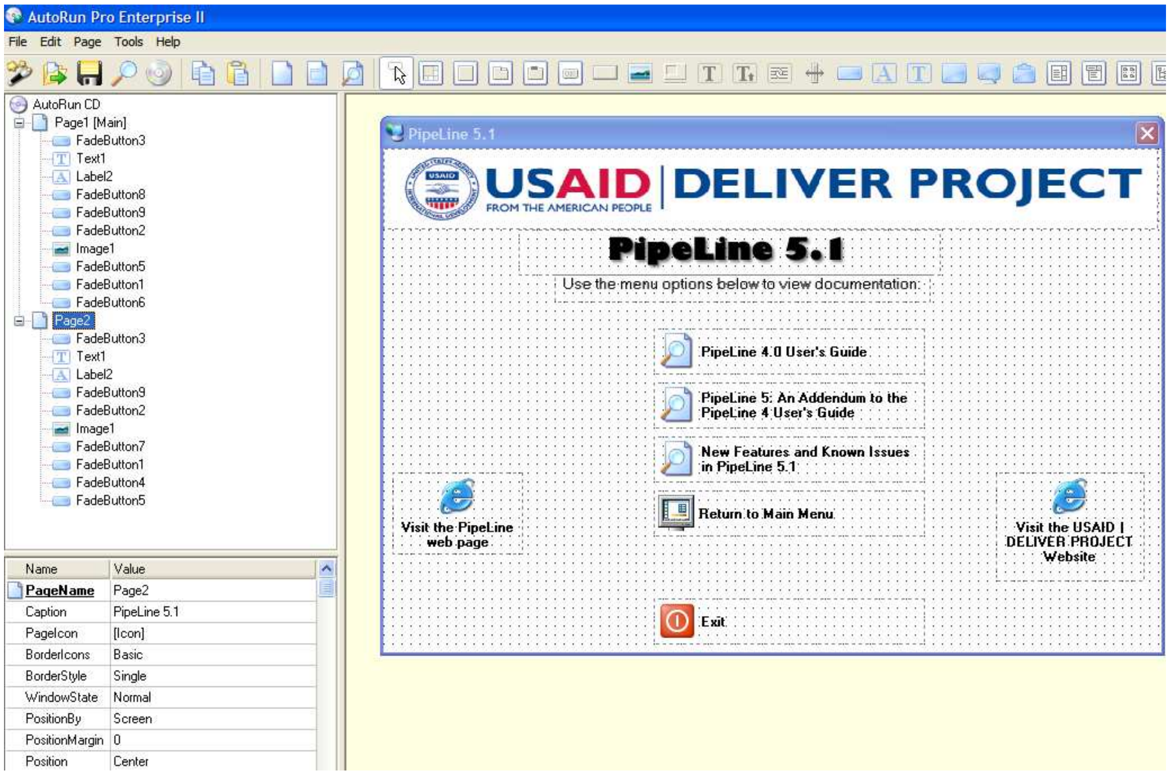
Action Type	Settings
NextPage	

Action Type: NextPage

Create second page

Using Page > Duplicate Page menu option second page was created.

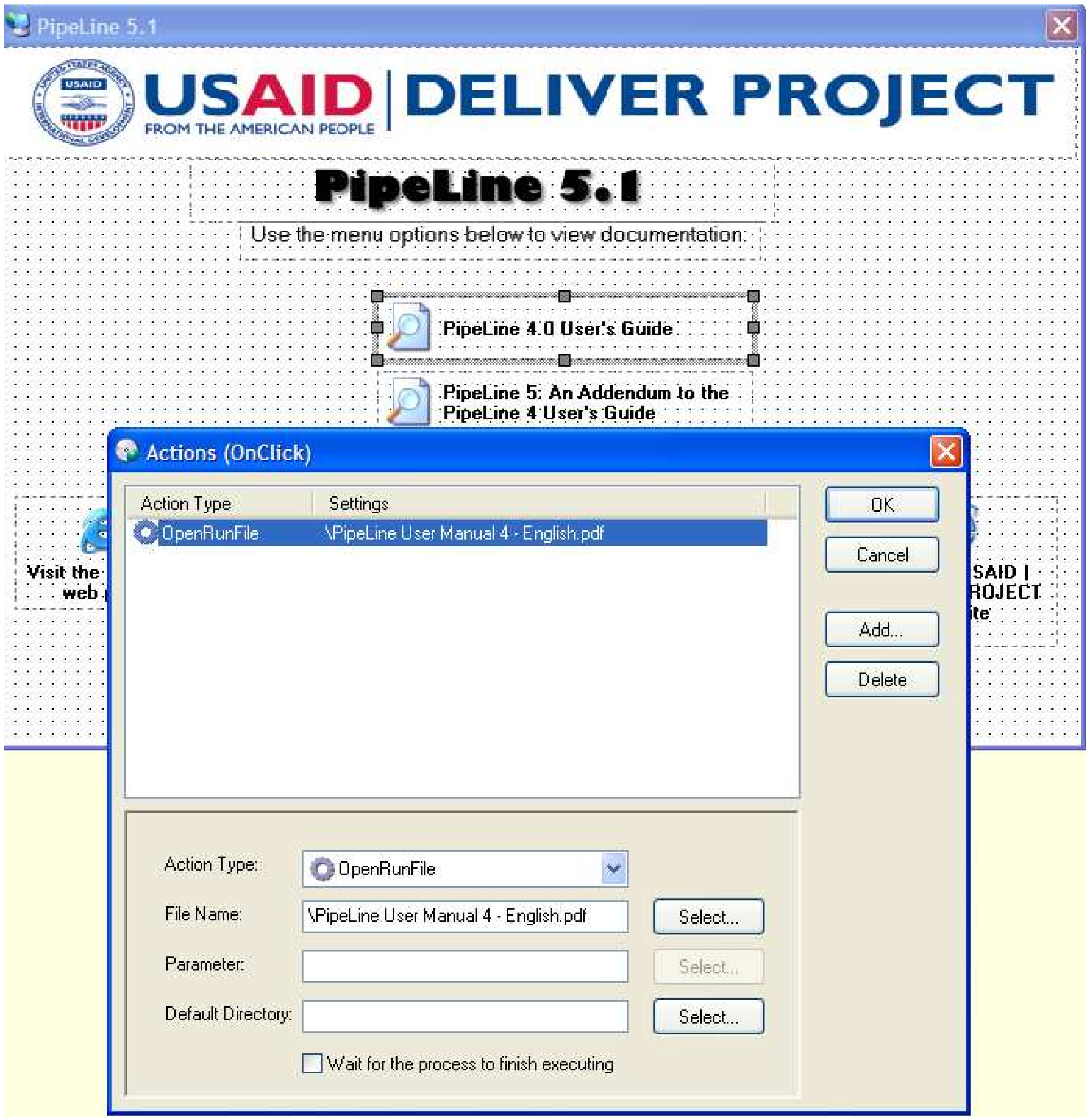
Figure 27 - Duplicate Page



Name	Value
PageName	Page2
Caption	PipeLine 5.1
PageIcon	[Icon]
BorderIcons	Basic
BorderStyle	Single
WindowState	Normal
PositionBy	Screen
PositionMargin	0
Position	Center

Create open PipeLine User Guide PDF file in Adobe Acrobat
Using OnClick > OpenRunFile click event, the open PDF file was defined.

Figure 28 - User Guide OnClick Action



Action Type	Settings
OpenRunFile	\PipeLine User Manual 4 - English.pdf

Action Type: OpenRunFile

File Name: \PipeLine User Manual 4 - English.pdf

Parameter:

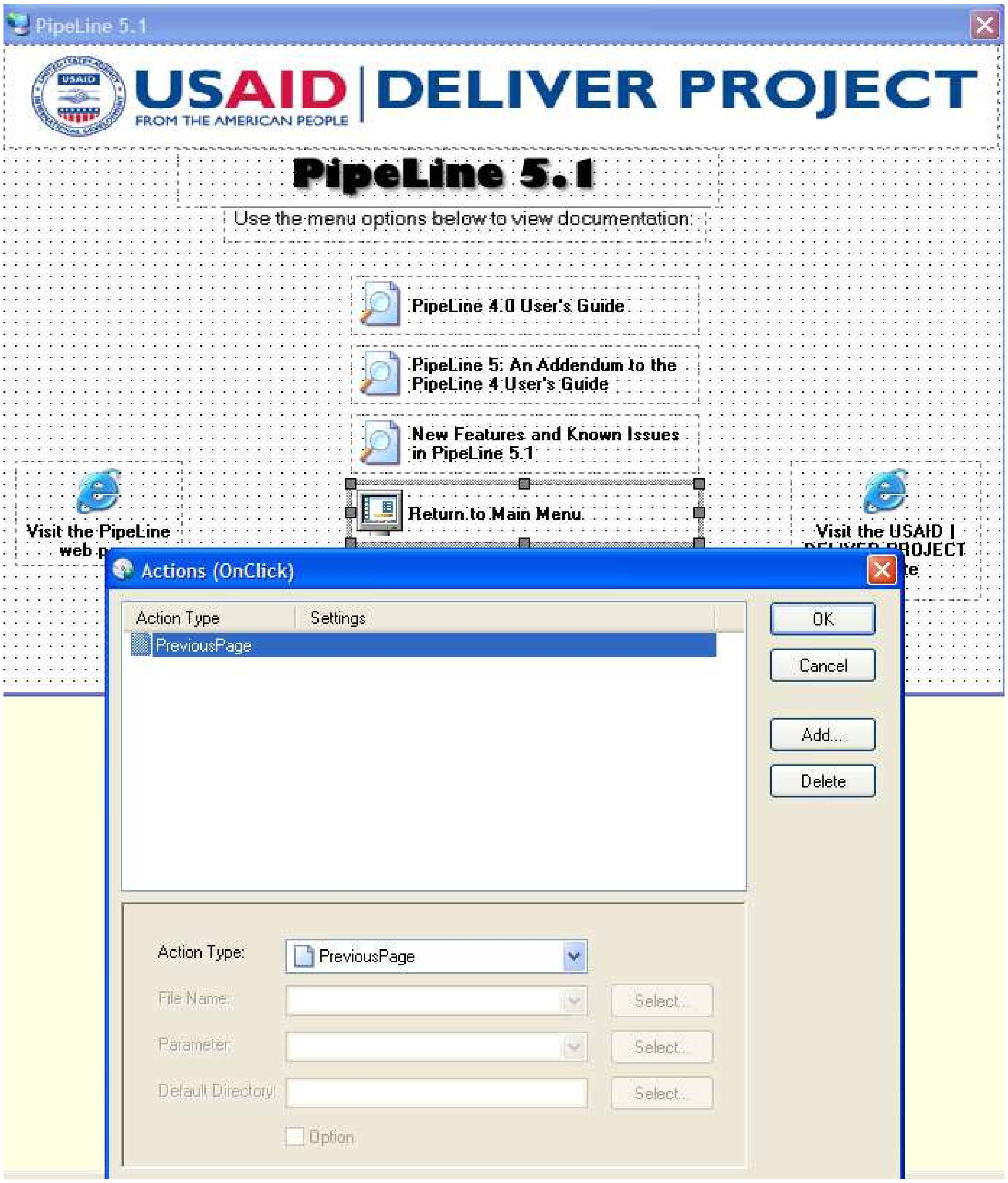
Default Directory:

☐ Wait for the process to finish executing

Create Return to Main Menu

Using OnClick > PreviousPage click event, return to previous page was defined.

Figure 29 - Return to Main Menu OnClick Action


PipeLine 5.1
USAID | DELIVER PROJECT
FROM THE AMERICAN PEOPLE
PipeLine 5.1
Use the menu options below to view documentation:
PipeLine 4.0 User's Guide
PipeLine 5: An Addendum to the PipeLine 4 User's Guide
New Features and Known Issues in PipeLine 5.1
Return to Main Menu
Visit the PipeLine web p
Visit the USAID | DELIVER PROJECT
Actions (OnClick)
OK
Cancel
Add...
Delete
Select...
Select...
Select...

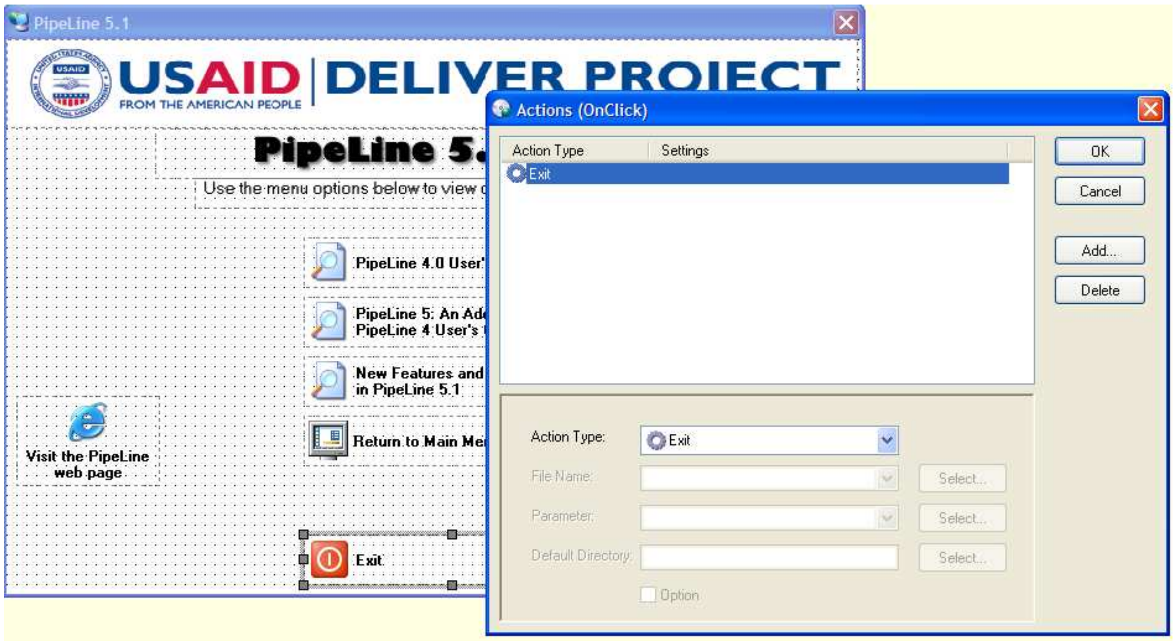
Action Type	Settings
PreviousPage	

Action Type: PreviousPage
File Name:
Parameter:
Default Directory:
Option

Create Exit menu option

Using OnClick > Exit click event, exiting the CD Run program was defined.

Figure 30 - Exit OnClick Action



Action Type	Settings
Exit	

Action Type: Exit
File Name:
Parameter:
Default Directory:
Option

Build the Executable AutoRun program and create the master CD

After the full two page menu choices were defined, it is time to test the project. Using File > Save and Test menu option, test the Auto install program. Once every element is tested and found acceptable, build the final version using File > Save and Publish menu option.

Figure 31 - File Dropdown Menu

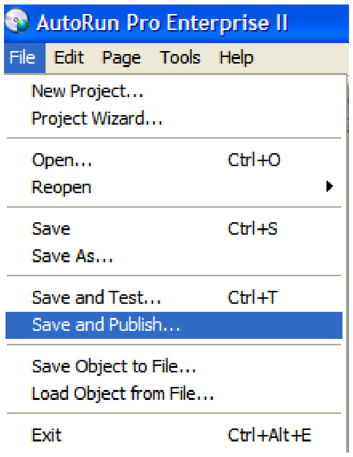
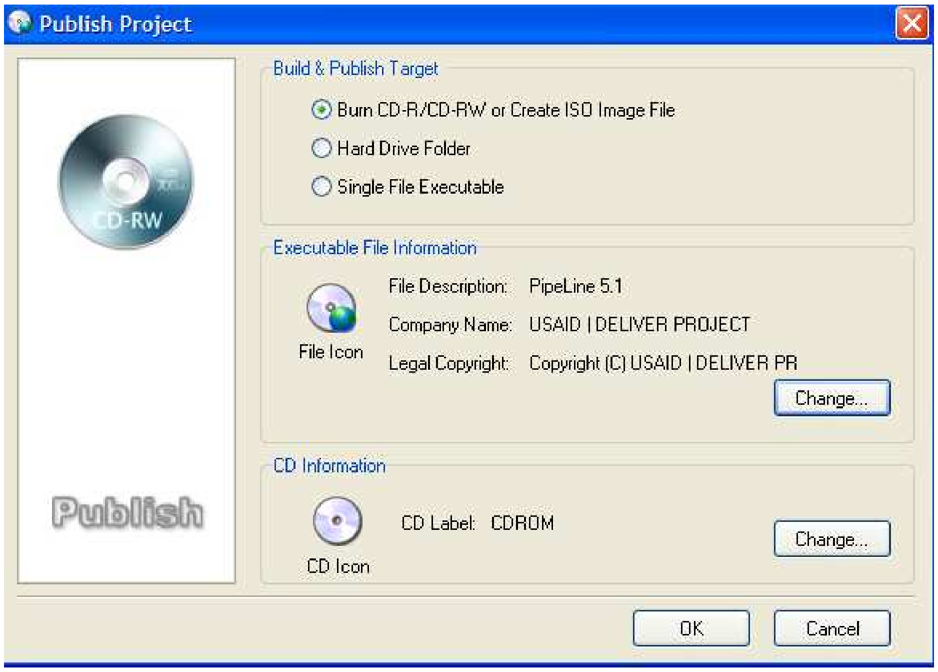


Figure 32 - Publish Project


Publish Project

Build & Publish Target

- [✓ Burn CD-R/CD-RW or Create ISO Image File]
- Hard Drive Folder
- Single File Executable

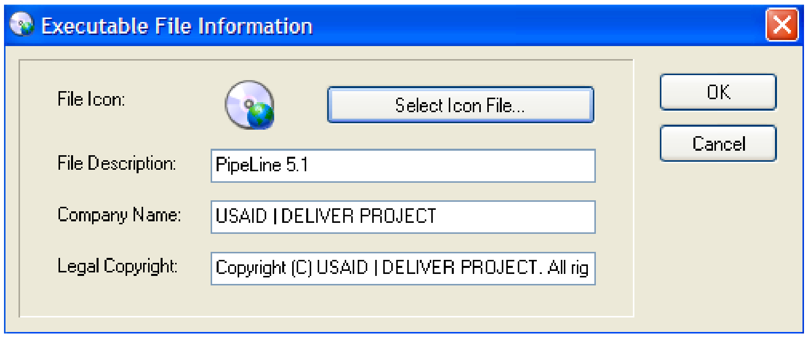
Executable File Information

File Description: PipeLine 5.1
Company Name: USAID | DELIVER PROJECT
Legal Copyright: Copyright (C) USAID | DELIVER PR

CD Information

CD Label: CDROM

Figure 33 - Executable File Information


Executable File Information

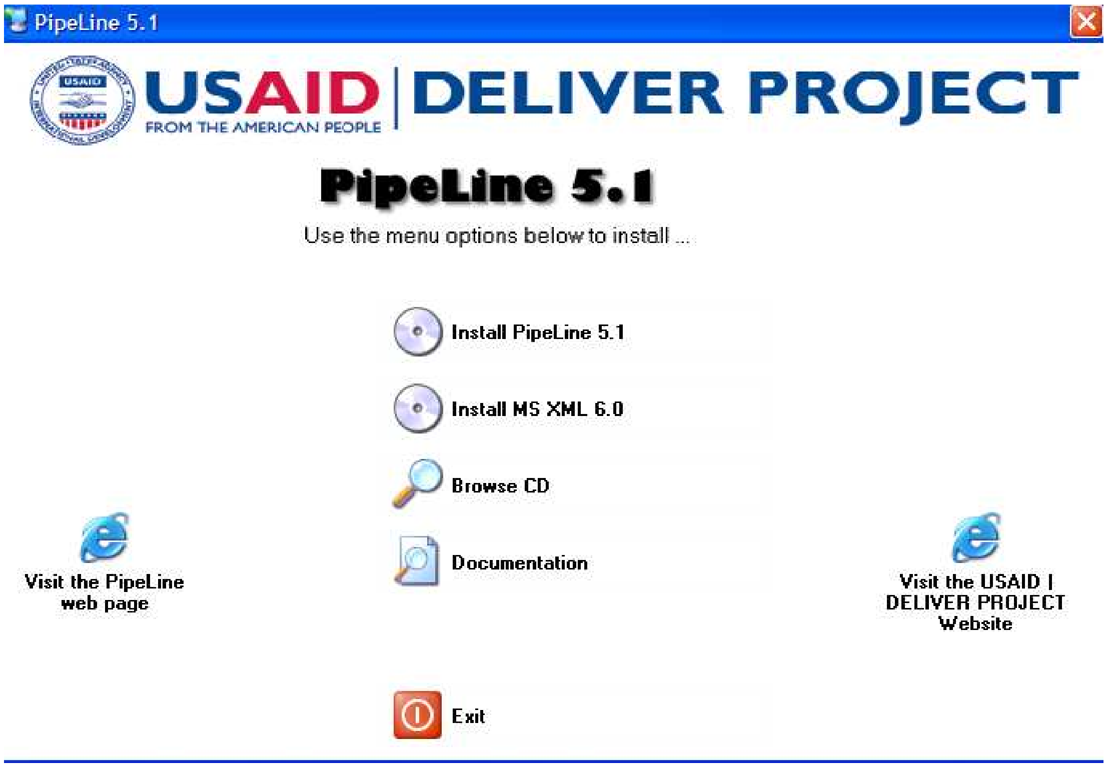
File Icon:
File Description: PipeLine 5.1
Company Name: USAID | DELIVER PROJECT
Legal Copyright: Copyright (C) USAID | DELIVER PROJECT. All rig

Test the final master CD and make required number of CD copies

After the master CD is build, test the CD under following operating environments:

- Windows XP
- Windows Vista
- Windows 7 32 bit
- Windows 7 64 bit

Figure 34 - Final Auto-run CD



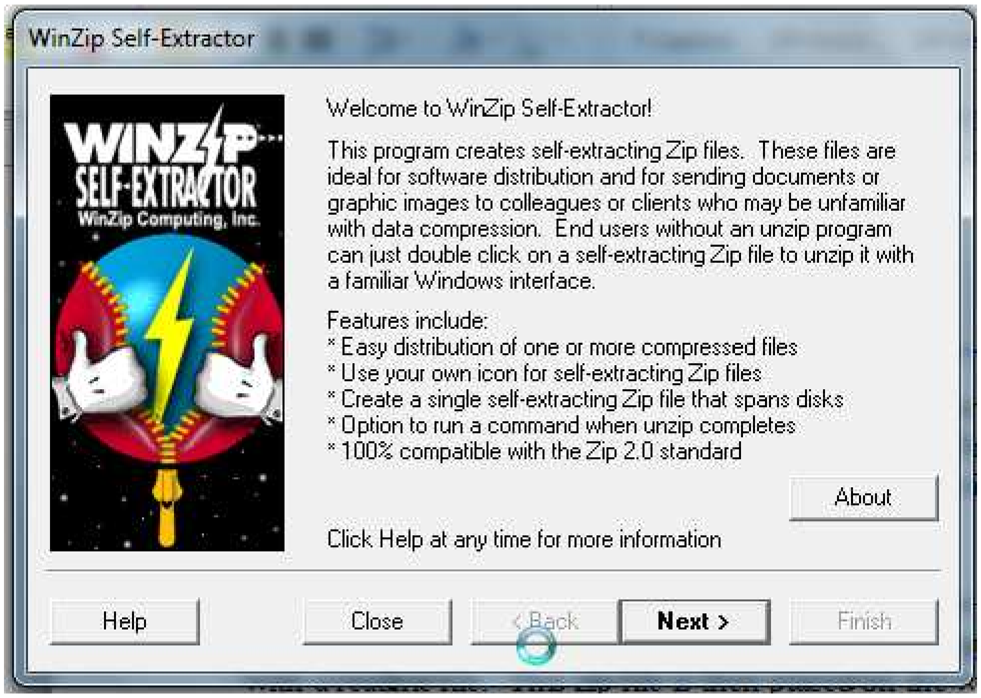
Final version of the Auto-run CD should look like above. After testing and acceptance, using a mass CD burner, publish required number of CDs. For professional quality CD label and high speed publishing, contact professional firms; provide master CD and related instructions.

Create Web Installer

Once the Auto run CD is created. The contents of the CD need to be packaged for distribution on the web. For this distribution, a self extracting exe file is created and zipped with a readme file. This zip file is then placed on the website for download. To create this self extracting exe file:

1. Using WinZip, zip all files found on the cd into a file PipeLine_master.zip.
2. Add the password, 374949449384, to this zip file.
3. Open WinZip Self Extractor 2.2 and click next.

Figure 35 - WinZip Self Extractor Welcome Window



Welcome to WinZip Self-Extractor!

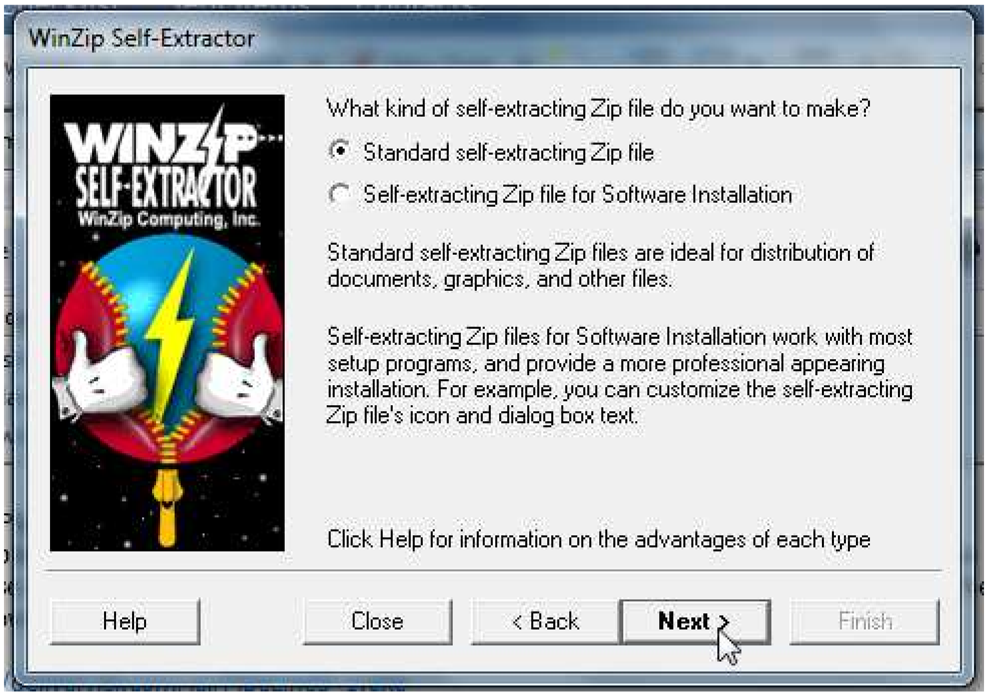
This program creates self-extracting Zip files. These files are ideal for software distribution and for sending documents or graphic images to colleagues or clients who may be unfamiliar with data compression. End users without an unzip program can just double click on a self-extracting Zip file to unzip it with a familiar Windows interface.

Features include:
* Easy distribution of one or more compressed files
* Use your own icon for self-extracting Zip files
* Create a single self-extracting Zip file that spans disks
* Option to run a command when unzip completes
* 100% compatible with the Zip 2.0 standard

Click Help at any time for more information

4. Select the option to create a standard self extracting zip file and click next.

Figure 36 - WinZip Self Extractor Select Type Window



What kind of self-extracting Zip file do you want to make?

[✓ Standard self-extracting Zip file]
Self-extracting Zip file for Software Installation

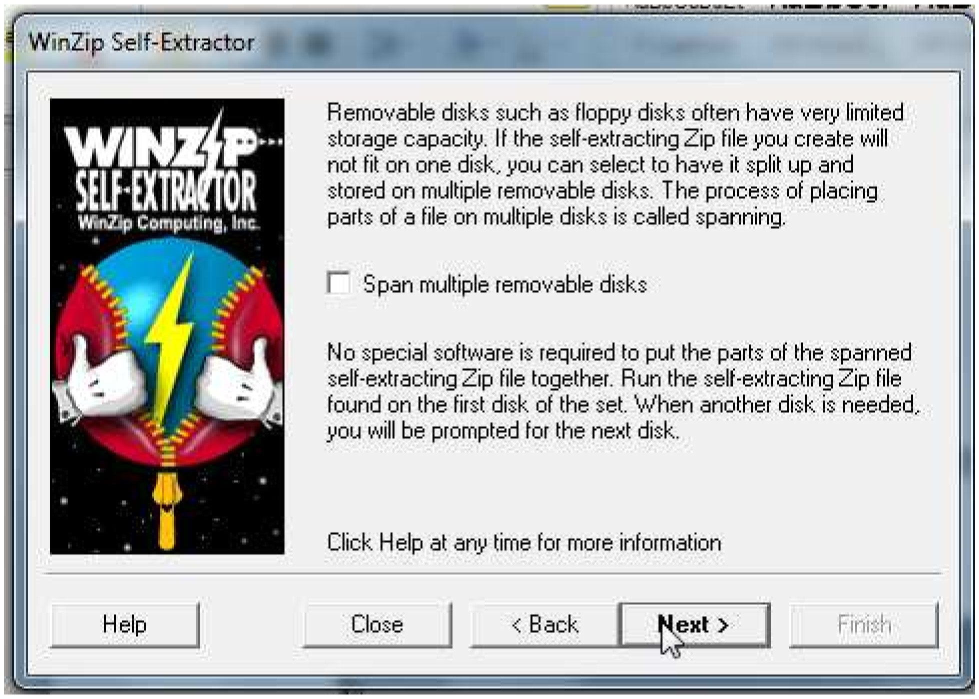
Standard self-extracting Zip files are ideal for distribution of documents, graphics, and other files.

Self-extracting Zip files for Software Installation work with most setup programs, and provide a more professional appearing installation. For example, you can customize the self-extracting Zip file's icon and dialog box text.

Click Help for information on the advantages of each type

5. Uncheck the option to span multiple removable disks and click next.

Figure 37 - WinZip Self Extractor Span Options Window


WinZip Self-Extractor

Removable disks such as floppy disks often have very limited storage capacity. If the self-extracting Zip file you create will not fit on one disk, you can select to have it split up and stored on multiple removable disks. The process of placing parts of a file on multiple disks is called spanning.

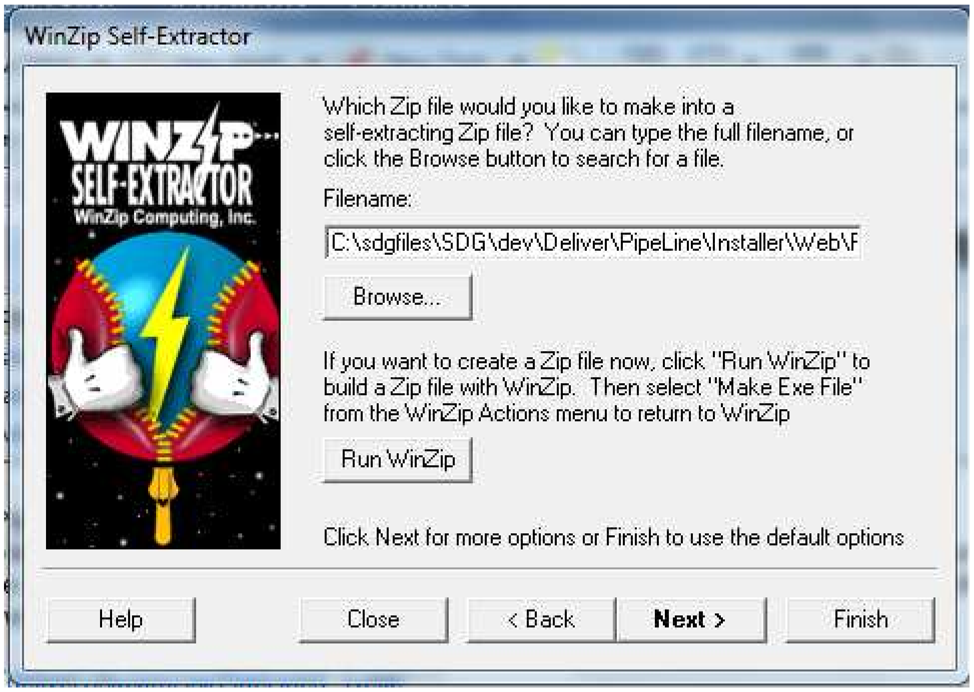
☐ Span multiple removable disks

No special software is required to put the parts of the spanned self-extracting Zip file together. Run the self-extracting Zip file found on the first disk of the set. When another disk is needed, you will be prompted for the next disk.

Click Help at any time for more information

6. Click browse and choose the master zip file and click next.

Figure 38 - WinZip Self Extractor File Selection Window


WinZip Self-Extractor

Which Zip file would you like to make into a self-extracting Zip file? You can type the full filename, or click the Browse button to search for a file.

Filename:

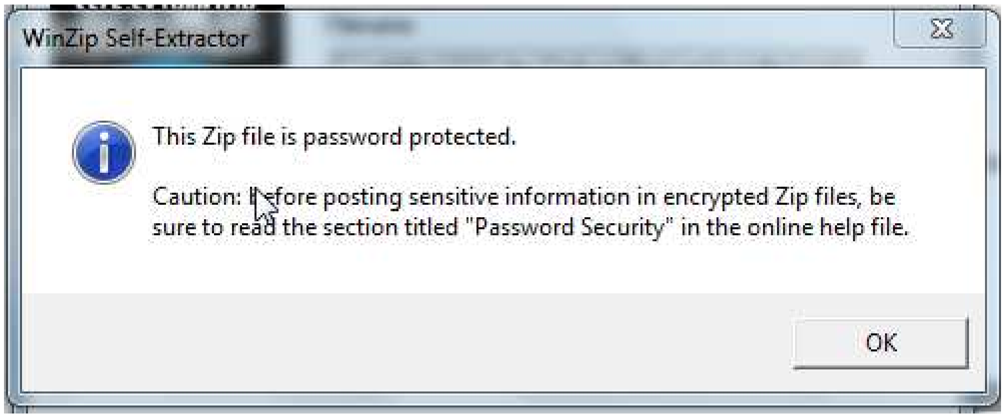
C:\sdgfiles\SDG\dev\Deliver\PipeLine\Installer\Web\F

If you want to create a Zip file now, click "Run WinZip" to build a Zip file with WinZip. Then select "Make Exe File" from the WinZip Actions menu to return to WinZip

Click Next for more options or Finish to use the default options

7. The following warning should appear and click OK.

Figure 39 - WinZip Self Extractor Password Notification



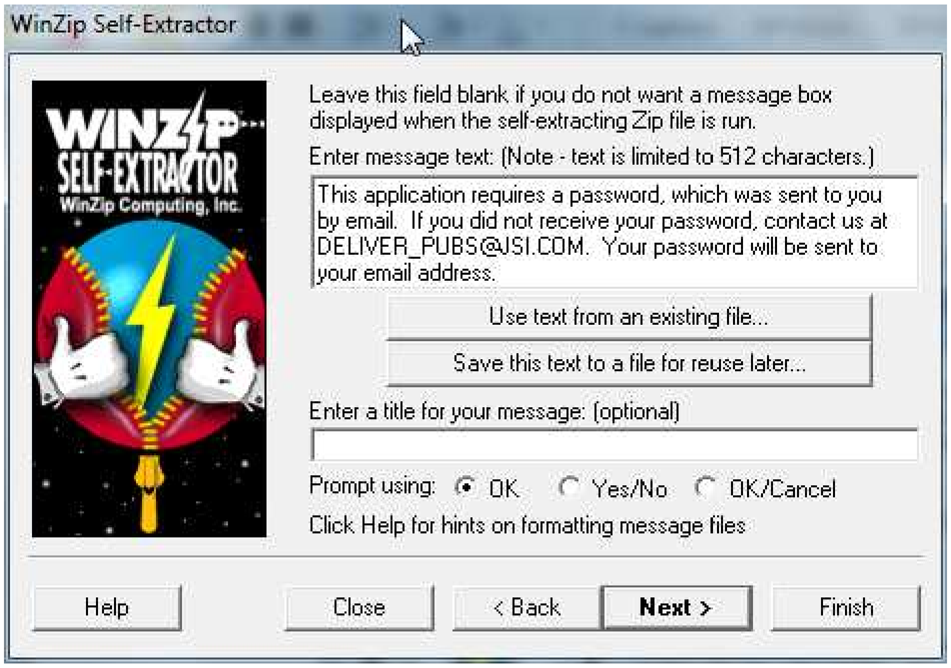
WinZip Self-Extractor

This Zip file is password protected.

Caution: before posting sensitive information in encrypted Zip files, be sure to read the section titled "Password Security" in the online help file.

8. Click "Use test from an existing file" and select the password.txt file found in the installer folder and click next.

Figure 40 - WinZip Self Extractor Message Text Option Window



WinZip Self-Extractor

Leave this field blank if you do not want a message box displayed when the self-extracting Zip file is run.

Enter message text: (Note - text is limited to 512 characters.)

This application requires a password, which was sent to you by email. If you did not receive your password, contact us at DELIVER_PUBS@JSI.COM. Your password will be sent to your email address.

Enter a title for your message: (optional)

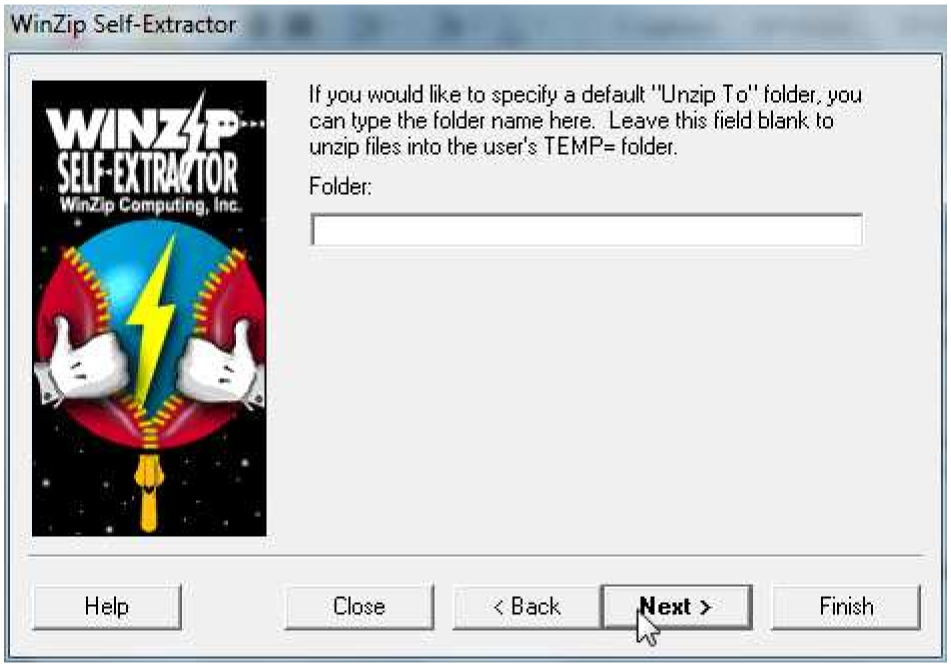
Prompt using: [✓ OK] Yes/No OK/Cancel

Click Help for hints on formatting message files

WINZIP SELF-EXTRACTOR
WinZip Computing, Inc.
Use text from an existing file...
Save this text to a file for reuse later...
Help
Close
< Back
Next >
Finish

9. Leave folder field blank and click next.

Figure 41 - WinZip Self Extractor "Unzip To" Option Window

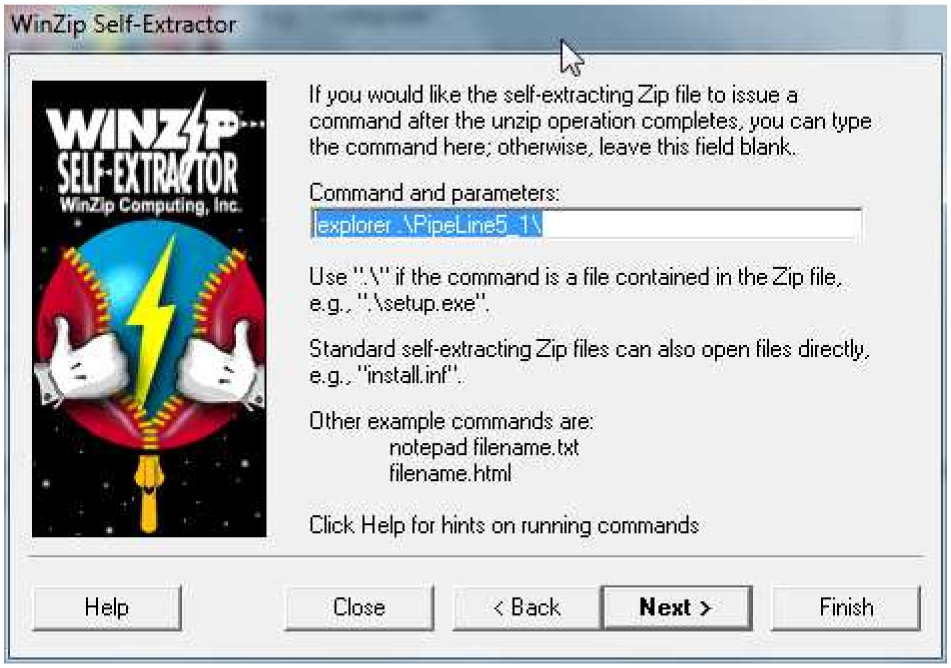

WinZip Self-Extractor

If you would like to specify a default "Unzip To" folder, you can type the folder name here. Leave this field blank to unzip files into the user's TEMP= folder.

Folder:

10. Enter value in command and parameters field as shown below and click next.

Figure 42 - WinZip Self Extractor Command Options Window


WinZip Self-Extractor

If you would like the self-extracting Zip file to issue a command after the unzip operation completes, you can type the command here; otherwise, leave this field blank.

Command and parameters:
explorer .\PipeLine5_1\

Use ".\" if the command is a file contained in the Zip file, e.g., ".\setup.exe".

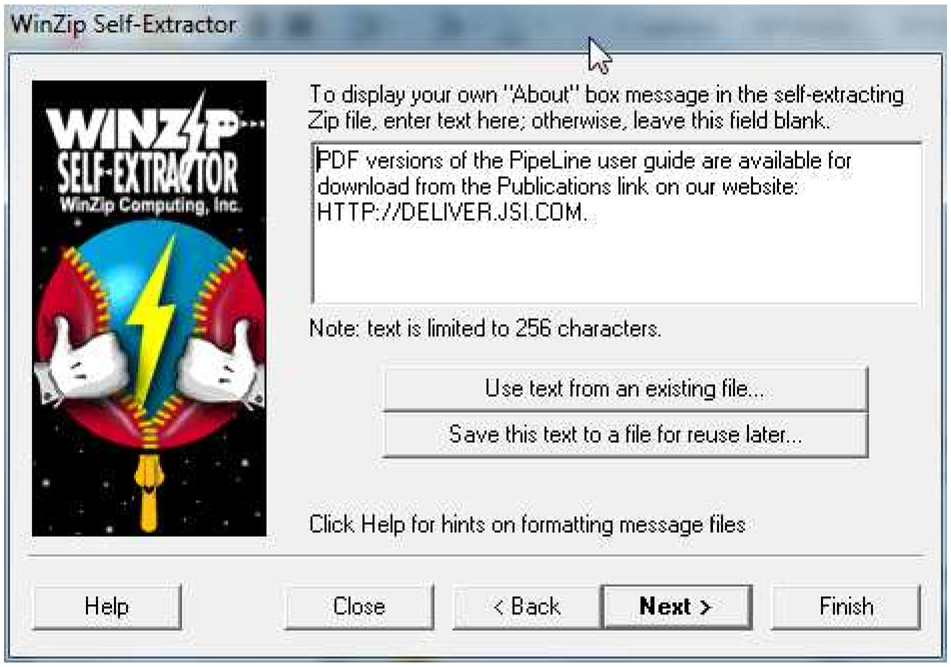
Standard self-extracting Zip files can also open files directly, e.g., "install.inf".

Other example commands are:
notepad filename.txt
filename.html

Click Help for hints on running commands

11. Click "Use test from an existing file" and select the about.txt file found in the installer folder and click next.

Figure 43 - WinZip Self Extractor About Box Options Window


WinZip Self-Extractor

To display your own "About" box message in the self-extracting Zip file, enter text here; otherwise, leave this field blank.

PDF versions of the PipeLine user guide are available for download from the Publications link on our website: HTTP://DELIVER.JSI.COM.

Note: text is limited to 256 characters.

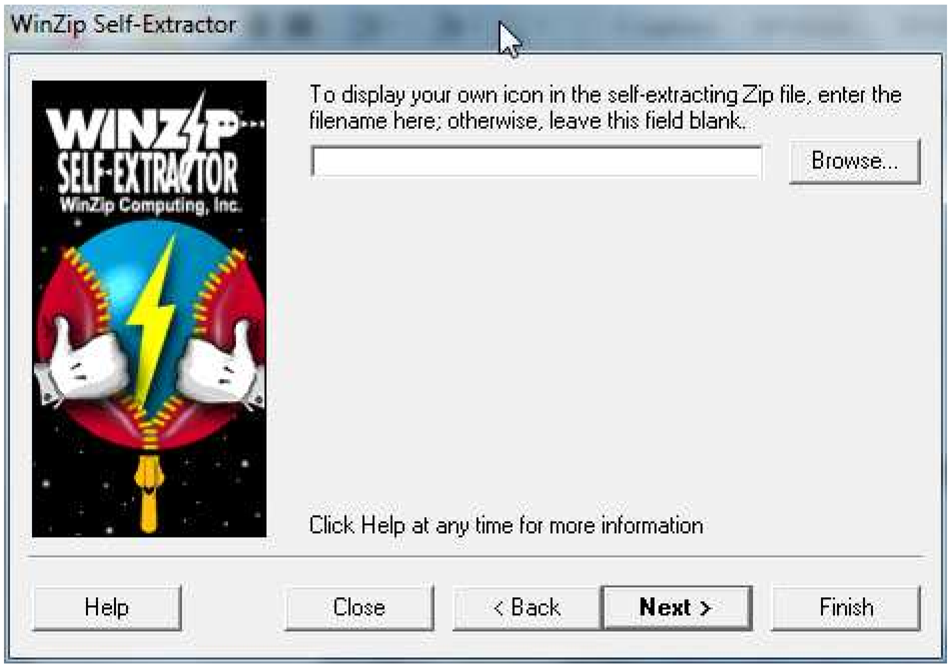
Use text from an existing file...
Save this text to a file for reuse later...

Click Help for hints on formatting message files

Help
Close
< Back
Next >
Finish

12. Leave the icon file name blank and click next.

Figure 44 - WinZip Self Extractor Icon Selection Window

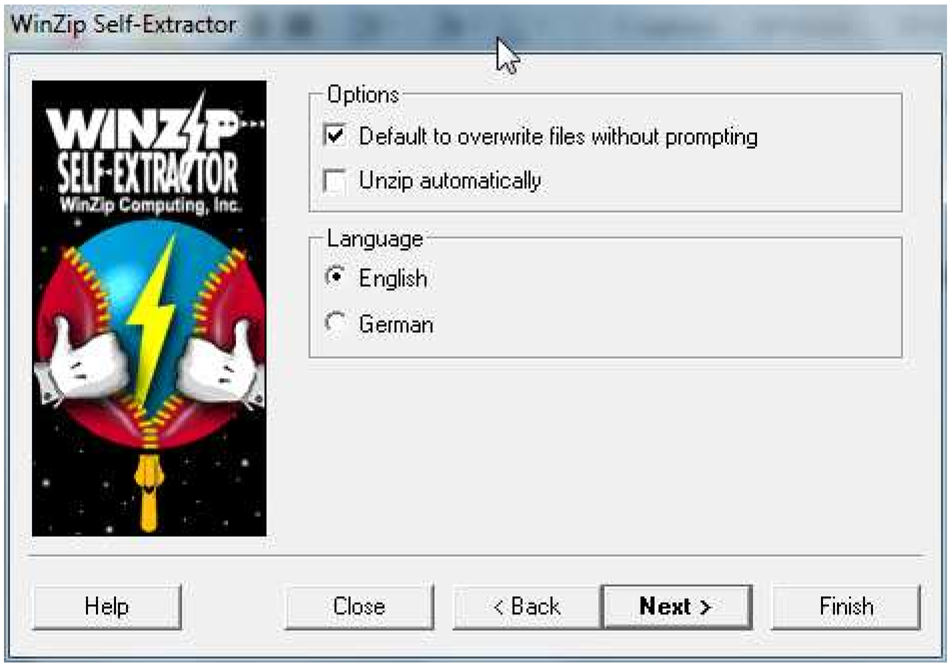

WinZip Self-Extractor

To display your own icon in the self-extracting Zip file, enter the filename here; otherwise, leave this field blank.

Click Help at any time for more information

13. Choose the options specified below and click next.

Figure 45 - WinZip Self Extractor Miscellaneous Options Window



WinZip Self-Extractor

Options

[✓] Default to overwrite files without prompting

[ ] Unzip automatically

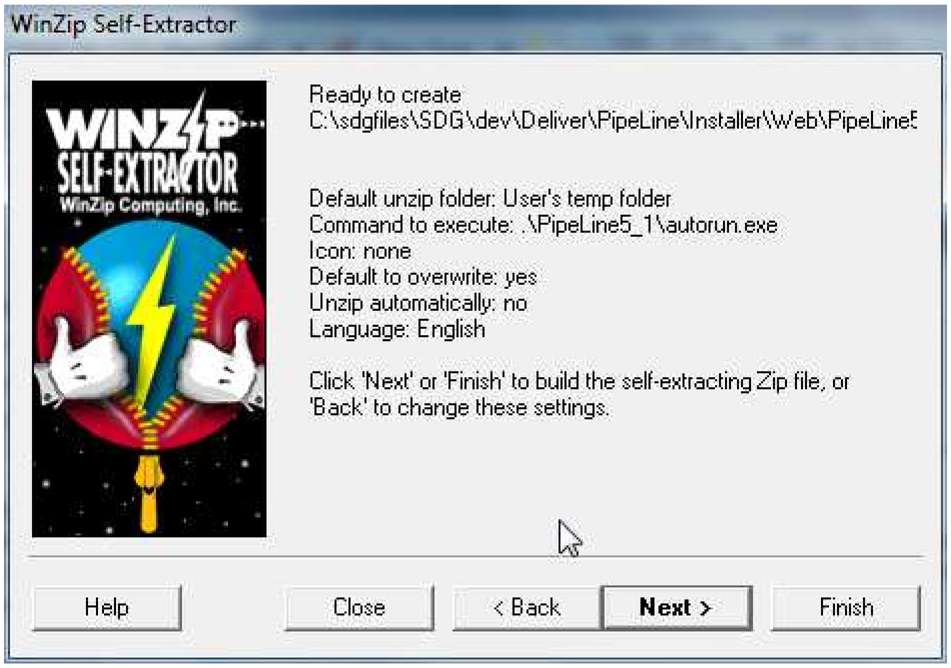
Language

(●) English

( ) German

14. Click Next to create the exe file.

Figure 46 - WinZip Self Extractor Ready to Create Window



WinZip Self-Extractor

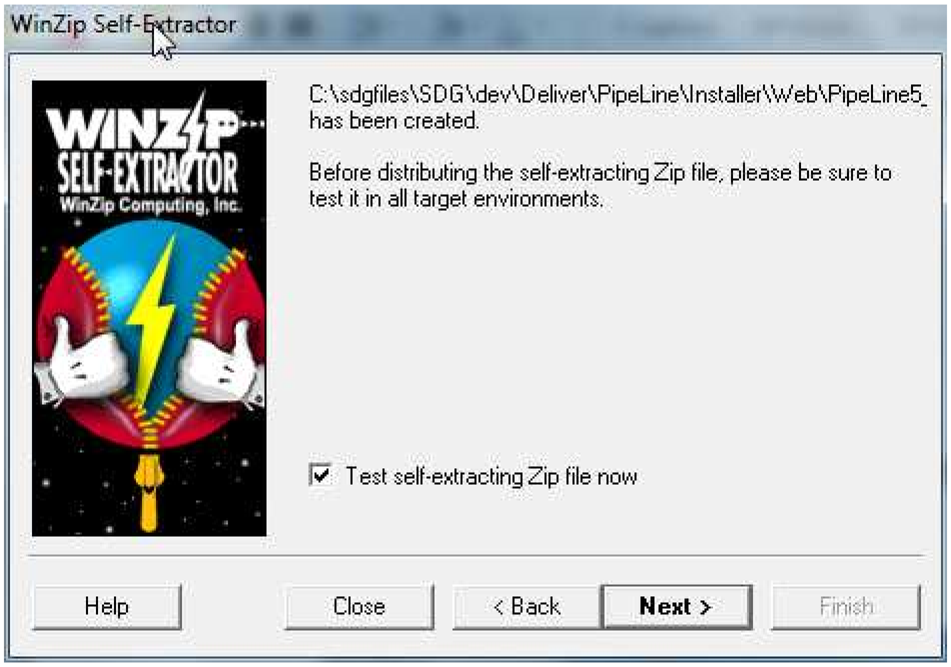
Ready to create
C:\sdgfiles\SDG\dev\Deliver\PipeLine\Installer\Web\PipeLine5

Default unzip folder: User's temp folder
Command to execute: .\PipeLine5_1\autorun.exe
Icon: none
Default to overwrite: yes
Unzip automatically: no
Language: English

Click 'Next' or 'Finish' to build the self-extracting Zip file, or
'Back' to change these settings.

15. When file is done being created, click next to text the file.

Figure 47 - WinZip Self Extractor Test Window


WinZip Self-Extractor

C:\sdgfiles\SDG\dev\Deliver\PipeLine\Installer\Web\PipeLine5_
has been created.

Before distributing the self-extracting Zip file, please be sure to test it in all target environments.

☑ Test self-extracting Zip file now

16. After testing file, select No, I am Finished and click Exit close WinZip Self Extractor.
17. Go to installation folder and rename exe file to PipeLine5_1.exe.
18. Select this file and the readme.txt file and create a new zip file named PipeLine5_1.zip. This is the file to be distributed on the web.

Web-download

The full installer is downloadable from the SDG website (http://sdg.jsi.com). Upon downloading, user will getbe requested to fill out a information form. The password for unzipping the web installer will then be emailed to the address provided on this form.

Application Design

Overview

Access stores all database tables, queries, forms, reports, macros, and modules in the Access Jet database as a single file.

For query development, Access offers a "Query Designer", a graphical user interface that allows users to build queries without knowledge of the SQL programming language. In the Query Designer, users can "show" the data sources of the query (which can be tables or queries) and select the fields they want returned by clicking and dragging them into the grid. One can set up joins by clicking and dragging fields in tables to fields in other tables. Access allows users to view and manipulate the SQL code if desired. Any Access table, including linked tables from different data sources, can be used in a query.

When developing forms and reports that are linked to queries placing or moving items in the design view, Access runs the linked query in the background on any placement or movement of an item in that Form or Report. If the form or report is linked to a query that takes a long time to return records this means having to wait until the query has run before you can add/edit or move the next item in the form or report (this feature cannot be turned off).

Non-programmers can use the macro feature to automate simple tasks through a series of drop-down selections. Macros allow users to easily chain commands together such as running queries, importing or exporting data, opening and closing forms, previewing and printing reports, etc. Macros support basic logic (IF-conditions) and the ability to call other macros. Macros can also contain sub-macros which are similar to subroutines. Macros however, are limited in their functionality by a lack of programming loops and of advanced coding logic. PipeLine only uses macros for populating the custom menubar. This allows for ease in creating the menubar at runtime and in opening the proper dialog box.

The programming language available in Access is, as in other products of the Microsoft Office suite, Microsoft Visual Basic for Applications, which is nearly identical to Visual Basic 6.0 (VB6). VBA code can be stored in modules and code behind forms and reports. Modules can also be classes.

To manipulate data in tables and queries in VBA, Microsoft provides two database access libraries of COM components:

- Data Access Objects (DAO) (32-bit only), which is included in Access and Windows
- ActiveX Data Objects ActiveX Data Objects (ADO) (both 32-bit and 64-bit versions)

PipeLine uses DAO objects and so DAO must be a registered reference for the application.

Many Access developers use the Reddick naming convention, though this is not universal; it is a programming convention, not a DBMS-enforced rule. It is particularly helpful in VBA where references to object names may not indicate its data type (e.g. tbl for tables, qry for queries).

Split Database Architecture

Microsoft Access applications, like PipeLine, adopt a split-database architecture. The database is divided into a front-end database that contains the application objects (queries, forms, reports, macros, and modules), and is linked to tables stored in a back-end shared database containing the data. The "back-end" database can be stored in a location shared by many users, such as a file server. The "front-end" database is distributed to each user's desktop and linked to the shared database. Using this design, each user has a copy of Microsoft Access installed on their machine along with their application database. This reduces network traffic since the application is not retrieved for each use, and allows the front-end database to contain tables with data that is private to each user for storing settings or temporary data. This split-database design also allows development of the application independent of the data. When a new version is ready, the front-end database is replaced without impacting the data database.

Linked tables in Access use absolute paths rather than relative paths, so the development environment either has to have the same path as the production environment or a "dynamic-linker" routine can be written in VBA. In PipeLine, the links to the backend are stored in the ProgV4.mdb file. This also allows us to populate the Window menubar option.

This is not an economical setup across slow networks, or in large organizations separated by great distances, as it will result in excessive lag to database users. Therefore, when users are installing PipeLine, consideration needs to be made for their network speed.

Naming Conventions

The conventions that follow were developed to promote uniformity and consistency in naming various program modules. Variables and objects with familiar labels make source code easier to read.

All MSAccess databases should utilize Reddick VBA (RVBA), Version 6.01 (see Appendix C). Table 5 summarizes the conventions used from the RVBA for database window objects.

Conventions used

Naming and coding conventions were not originally used. With version 3.0, these conventions have been used. All future programming should follow these guidelines.

Table 5 – Prefixes for Access Database Window Objects

Object	Prefix	Description
Table	tbl	Data Table
	tmp	Temporary Table
Query	qsel	Select Query
	qupd	Update Query
	QDEL	Delete Query
	qapp	Append Query
FORMS	frm	Form
	fsub	Subform
Reports	rpt	Report
	rsub	Subreport
Macros	mcr	Macro
Modules	bas	Module

Table 6 summarizes the conventions for object types found in form and reports.

Table 6 – Access Object Variable Prefixes

Prefix	Object Type
chk	Checkbox
cbo	Combobox
cmd	Command Button
lbl	Label
lst	listbox
ole	ObjectFrame
opt	OptionButton
pge	Page
Tab	Tab Control
Txt	Text Box
Tgl	Toggle Button

A prefix should first be added to each field in a table reflecting a two-digit abbreviation of the table name followed by an underscore and the descriptive naming convention for the object type. The first part of this prefix should reflect the table where it is located or the table name of the parent table if the field is a foreign key (i.e. ea_datUpdated could be included in a table named tblExpectedActions and fr_datReceived could be included in the same table thus showing its link to the parent table tblFundsReceived). Table 7 summarizes the naming conventions for the object types in the tables.

Table 7 – Field Name Conventions

Prefix	Object Type
bin	Binary
byt	Byte
lng	Long
cur	Currency
dat	Date/Time
dbl	Double
int	Integer
mem	Memo
sng	Single
str	Text
f	Yes/No

Data Types

The following table contains the data types in the database tables along with other common data formats.

Table 8 – Data Types

Type Field Size	Description	Format	Prefix
Yes/No	Used to setup fields containing boolean values. The default value is set to False.	True/False	f
Currency	Used to setup fields containing numeric values referring to US dollars.	2 decimal places	cur
Number			
Long Integer	Used to setup fields containing long integer values.		lng
Single	Used to setup fields containing single-precision floating-point values.		sng
Double	Used to setup fields containing double-precision floating-point values.		dbl
Byte	Used to setup fields containing byte values.		byt
Integer	Used to setup fields containing generic integer values.		int
Memo	Used to setup character fields spanning multiple lines.		mem
Date	Used to setup fields containing date values.	dd-mmm-yyyy	dat
Text 1 - 255	Used to setup fields containing generic single-line text values.		txt
Zip Codes 10	Used to setup fields containing zip codes.	00000\-9999;0;_	txt
Phone #/Fax # 16/10	Used to setup fields containing phone/fax numbers.	\(000") "000\-0000\ a#####;1;_	txt
Percent	Used to setup fields holding percent values.	2 decimal places	per

Graphical User Interface

The following conventions were developed to promote uniformity and consistency in appearances. This MSAccess database utilizes Microsoft Application User Interface Guidelines created by JSI. (See Appendix E).

Explorer-Style Navigation

Explorer-style navigation provides for easiest navigation of the application's screens. It contains a treeview list of all screens available in the application, a list view of the records available to view and the detail view of the selected record. (For Reports the list view region will display the parameters and the detail view will display the report results based on selected parameters.) The following drawing defines the recognized screen regions and their sizes.

Resolution

The standard screen resolution has been updated to 1024x768. Please note that when applying this upgrade, the tree view, list view and detail view regions were increased proportionally.

Figure 48 – Sample Menu

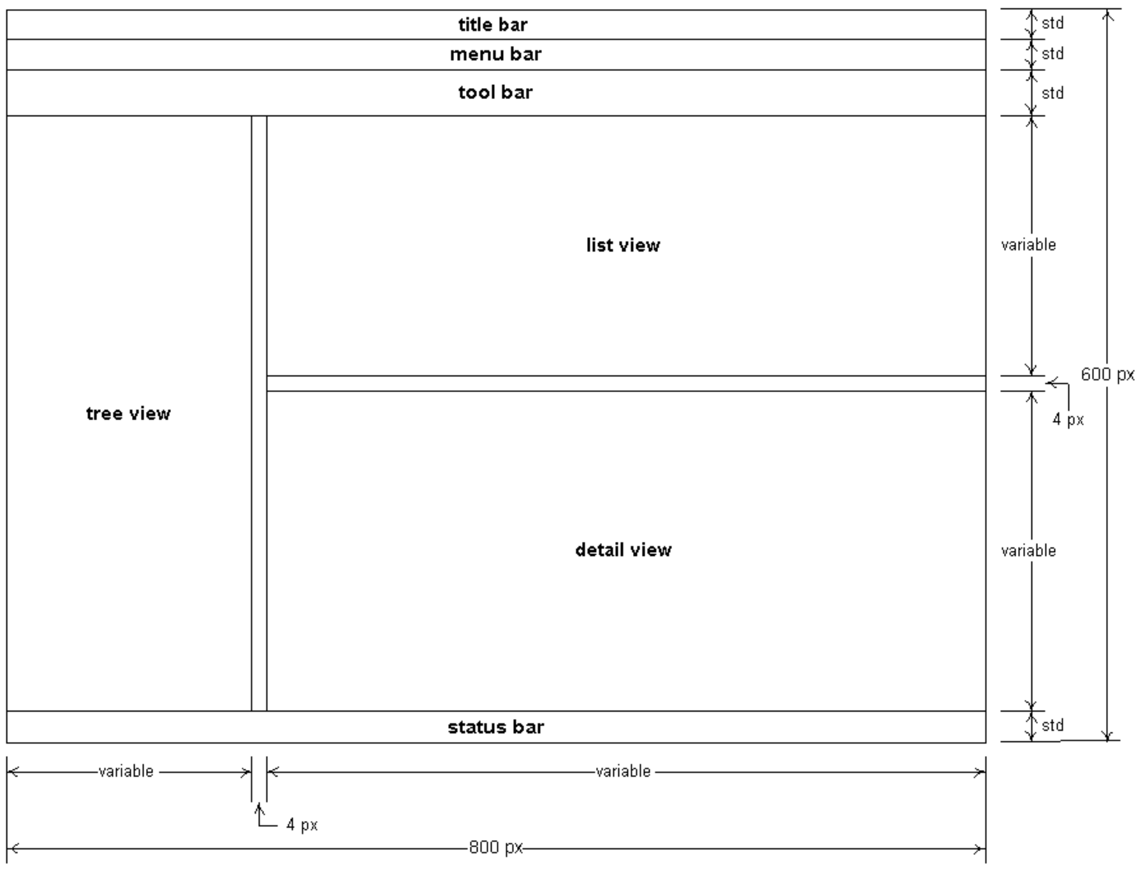


Table 9 – Menu Element Specification Table

Element	Comments
Title Bar	Should be set with the System Title and selected tree view title
Menu Bar	Should contain the minimum options needed for the form.
Tool Bar	(Optional) Should contain icons for quick access to various tools/features.
Tree View	On click should repopulate List View and Detail View with appropriate information.
List View	Should list the various record available in the detail view. In the case of reports, this should list the various parameters available for the report.
Detail View	Should display detailed information for the item selected in the list view or the parameters selected for the report.
Status Bar	Should provide quick simple explanation for the current field on the form.

Menu Bar Specification

In PipeLine the Menu Bar contains the following options.

Table 10 – Standard functionality of Switchboard Strip Menus

Main Choice	Sub Choice	Description
File	Exit	Close application and disconnect from the database
Import		
Export		
Tools		

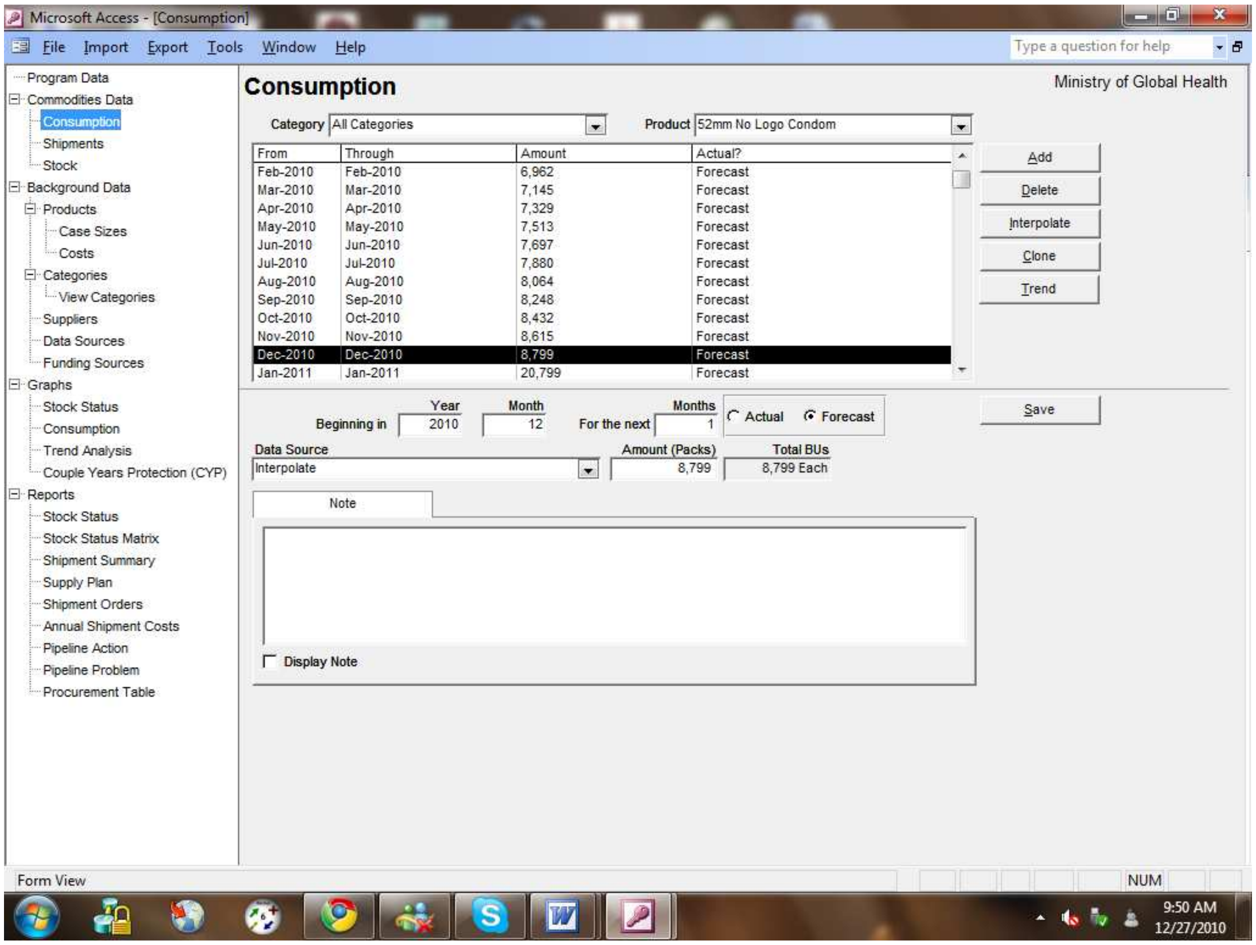
Main Choice	Sub Choice	Description
Window		
Help	Help	Open online help for application. If no online help has been developed, should open Access help.

Form Conventions

As shown in the figure below, the standard form lets users view record details and all updatable fields are enabled. Upon selecting an item in the list view, the form will filter to the correct record. (User may also select new to be taken to a new record where all fields are empty.)User may then click on SAVE to save the data. If the user modifies the data and tries to leave the form without clicking save, they will be prompted to save. To delete data from the application, the user will select the record in the list view and click the DELETE button. They will be prompted to verify the deletion.

The sample below depict standard for the list and detail screen.

Figure 49 – Sample Detail Form



Microsoft Access - [Consumption]

File Import Export Tools Window Help

Type a question for help

- Program Data
- Commodities Data
  - Consumption
  - Shipments
  - Stock
- Background Data
  - Products
    - Case Sizes
    - Costs
  - Categories
    - View Categories
  - Suppliers
  - Data Sources
  - Funding Sources
- Graphs
  - Stock Status
  - Consumption
  - Trend Analysis
  - Couple Years Protection (CYP)
- Reports
  - Stock Status
  - Stock Status Matrix
  - Shipment Summary
  - Supply Plan
  - Shipment Orders
  - Annual Shipment Costs
  - Pipeline Action
  - Pipeline Problem
  - Procurement Table

Consumption

Ministry of Global Health

Category: All Categories

Product: 52mm No Logo Condom

From	Through	Amount	Actual?
Feb-2010	Feb-2010	6,962	Forecast
Mar-2010	Mar-2010	7,145	Forecast
Apr-2010	Apr-2010	7,329	Forecast
May-2010	May-2010	7,513	Forecast
Jun-2010	Jun-2010	7,697	Forecast
Jul-2010	Jul-2010	7,880	Forecast
Aug-2010	Aug-2010	8,064	Forecast
Sep-2010	Sep-2010	8,248	Forecast
Oct-2010	Oct-2010	8,432	Forecast
Nov-2010	Nov-2010	8,615	Forecast
Dec-2010	Dec-2010	8,799	Forecast
Jan-2011	Jan-2011	20,799	Forecast

Add | Delete | Interpolate | Clone | Trend

Beginning in: Year 2010, Month 12

For the next: Months 1

Actual [✓ Forecast]

Save

Data Source: Interpolate

Amount (Packs): 8,799

Total BUs: 8,799 Each

Note

Display Note

Form View

NUM

9:50 AM
12/27/2010

For each control in PipeLine a specific style must be applied. This is controlled by the tblStyle table in the application. The tlkTranslationText table contains the labels for all controls on all forms in the application. In this table, a style form the tblStyle is selected. When opening a form, the styles and labels are then applied to the control. This allows the entire application to be uniform and if the style requirements change, the developer only needs to modify the style table to apply the change to all forms. Below is the list of styles for each element:

Table 11 - Element Styles

Control Type	Font	Font Size	Fore Color	Font Weight	Underline?	Special Effect	Border Style	Back Color
Title	Arial	14		700	No	Flat	transparent	
ProgramName	Arial	10		400	No	Flat	transparent	
Main Selection	Arial	10		400	No	Sunken	solid	
General Text	Arial	8		400	No	Sunken	solid	
RptTitle1	Arial	12		700	No	Flat	transparent	
RptHeader	Arial	8		400	No	Flat	transparent	
RptTitle2	Arial	10		400	No	Flat	transparent	
RptLabel	Arial	8		400	No	Flat	transparent	
Cmd Button	arial	8		400	No			
Checkbox					No	Sunken	solid	
TabCtrl	Arial	8		400	No	Sunken		
Frame					No	Sunken	solid	
graph					No	Sunken	solid	
Subform					No	Sunken	solid	
RptGeneralHead w/o Border	arial	8		400	No	Flat	transparent	
Hidden (fc=bc)	arial	8		400	No	Flat	solid	
RptGroup1	Arial	8		700	No	Flat	transparent	
RptGroup2	Arial	8		400	No	Flat	transparent	
RptGroup3	Arial	8		400	No	Flat	transparent	
RptGroup4	Arial	8		400	No	Flat	transparent	
General Label	Arial	8		400	No	Flat	transparent	
Main Select Label	Arial	10		400	No	Flat	transparent	
General sfr Text	Arial	8		400	No	Flat	solid	
RptGeneral w/Border	Arial	8		400	No	Flat	solid	
RptGeneral w/o Border	Arial	8		400	No	Flat	transparent	
RptGeneralHead w/Border	arial	8		700	No	Flat	solid	
RptHidden (fc=bc)	arial	8		400	No	Flat	transparent	10092543
MoveButton	arial	10		400	No			
Hypertext	Arial	8	255	400	No	Flat	transparent	
RptTitle2_Highlighted	Arial	10		400	No	Flat	transparent	10092543
RptGeneral w/Border Highlighted	Arial	8		400	No	Flat	solid	10092543

Control Use

MSAccess features several different types of standard controls. It's often possible to use two different controls to achieve the same general functionality. For example, a list of static values could be represented as a set of radio buttons or as pop list. The following guidelines should be used to determine what the best control for the job is.

Table 12 – Control rules

Control	#	Rules
Check Boxes	1	Check Boxes should always be used for True/False fields unless the True/False paradigm does not obviously apply in which case two Radio Buttons should be used
Combo Box	1	Combo Boxes should be used in cases where a list may change or exceeds 8 values
	2	The user should always select on a descriptive value rather than a code
List Box	1	List Boxes should be used in cases where a list may change and it's important for the user to be able see the other options
	2	The user should always select on a descriptive value rather than a code
	3	List Boxes should be used in cases where it's necessary for the user to keep track of multiple selected/ deselected options (i.e., Countries to include on a report)
Command Buttons	1	Command Buttons should be used to indicate a decision made by the user.
Radio Buttons	1	Radio Buttons should be used in cases where there are less than 8 choices and these choices are static.
	2	Text Labels should be placed to right of Radio Buttons
Tab	1	Tab canvases should be used to model forms that would exceed a single screen. Also, tab canvases can be used to incorporate affiliated processes into a single module.

Command Buttons

PipeLine contains thirteen standard command buttons that provide consistent functionality throughout the application. The buttons should only be used to provide the following functionality. Additional or different functionality should be provided by another command button.

When a command button is selected, the ON CLICK event procedure is fired, which calls the appropriate code. The table below describes the functionality of standard buttons.

Table 13 – Functionality of standard buttons

Button	Description
Add	Allow the user to enter a new record in the current recordset
Delete	Allow the user to delete the current record from the current recordset.
Save	Saves all changes to the current record since edit button was selected
Cancel	Cancels all changes to the current record since edit button was selected
Close	Closes form. Returns to calling form
Print	Sends the report directly to the printer.
Preview	Shows the print preview of the report.
PDF	Sends the report directly to the pdf printer specified.
ShowData	Shows data for report based on selected parameters in the detail view of the form.
Hide Data	Hides data for report in the detail view of the form.

Report Conventions

Report Layout

The conventions on the following pages are to be used when designing and laying out reports:

Table 14 – Report Checklist

Group	Items	Specification
General	Report Output	Should be horizontally centered on the page.
	Default Paper Size (8.25 x 11)	Reports should be created to fit on both A4 and Letter sized paper.
	Parameters	if parameters are required, they should be called from the list view section of the form so that the parameters can be validated.
Report Margins	Portrait Reports	.5" Top .75 Bottom .5" Left and Right
	Landscape Reports	.5" Top and Bottom .5" Left .75" Right
Report headings	Report header including system title, report title, any other centered columns, run date, run time, page number	Set by tblStyle
	1st line of the report title	Report Name
	2nd line of the report title	SubReport Name
	3rd line of the report title	Description of Report including filtering summary (if applicable)
	Report title spacing	Report titles should be stacked. There shouldn't be a line or half line between titles.
	Application Name, Report Display Name, and Program Name	Should be aligned with the left margin
	Application Name	PipeLine 5.1
	Report Display Name	As specified on Program Form
	Program Name	As specified on Program Form
	Run Date, Run Time, and Page Number	Should be aligned with the right margin
	Run Date	DD-MMM-YYYY Format
	Run Time	Format as short time
	Page Number	="Page " & [Page] & " of " & [Pages])
Report Body	Report body text	Set by tblStyle
	Report body labels	Set by tblStyle
Column Headings	Column headings	Justification of columns should match headings. Right justified amounts should have right justified headings, etc.
	Column headings	Should appear on every page - but only once.
Totals	Totals	Should not appear by themselves on a page.

Group	Items	Specification
	Total labels	Should appear to the left of the total row. The total label should be formatted bold and include the value of the group being totaled - i.e. Total Kenya or Total AVSC. Don't use "Subtotal".

Icons

The standard MSAccess icons should be used.

VBA Coding Guidelines

VBA can be difficult to read when coded badly. PipeLine follows the Reddick VBA (RVBA) Coding Conventions (see Appendix D). The following outlines the proper way to document and structure VBA program units.

Procedure/Function Declarations

If an event procedure is more than 10 lines of code or is used more than once in the system, then a module should be created containing that public function. Any subroutine of the function should be included in the module and should be private if not being called by any other function.

Variable Declarations

Variables should also follow the naming conventions outlined in this chapter.

Comments

Every event procedure and functions should have a header comment. The header comment should include the following:

The purpose of the procedure or function

The purpose and definition of input parameters

The definition of any returned parameter

The date created and name of the primary developer

The dates, initials and technical notes of any subsequent developers

The header comments should appear in the following format.

```
‘ Comments:
‘ Parameters:
‘ Returns:
‘ Created:
‘ Modified:
‘____________________-
```

Any section of code that needs further explanation or is based on an external assumption should be commented. These comments should appear in the following format.

```
 ‘ This is the comment
```

White Space

Wherever the readability of the code will be enhanced (i.e. Around IF statements, FOR Loops, etc.) developer’s are encouraged to use white space.

Indentation

Developers should follow standard indentation practices when writing code.

Global Variables

Unless absolutely necessary, Global Variables should not be used. Instead use passed parameters or form level parameters

Database Schema

Overview

PipeLine has two distinct types of data: Commodity data and Background data. The Entity Relationship Diagram below shows the relations ship of the commodities data, i.e. Shipments, Inventory, and Consumption, and their relationship with the Product Background Table. The other background tables describe the Product Table and/or the commodity data: The second diagram below shows the relationships between these tables. The remaining tables are not part of the ERD since they either contain reference data, for example the translations tables, which are used to store information for the application itself, or are temporary tables used to store data temporarily when viewing forms and reports.

Entity Relationship Diagrams

Figure 50 - ERD Diagram

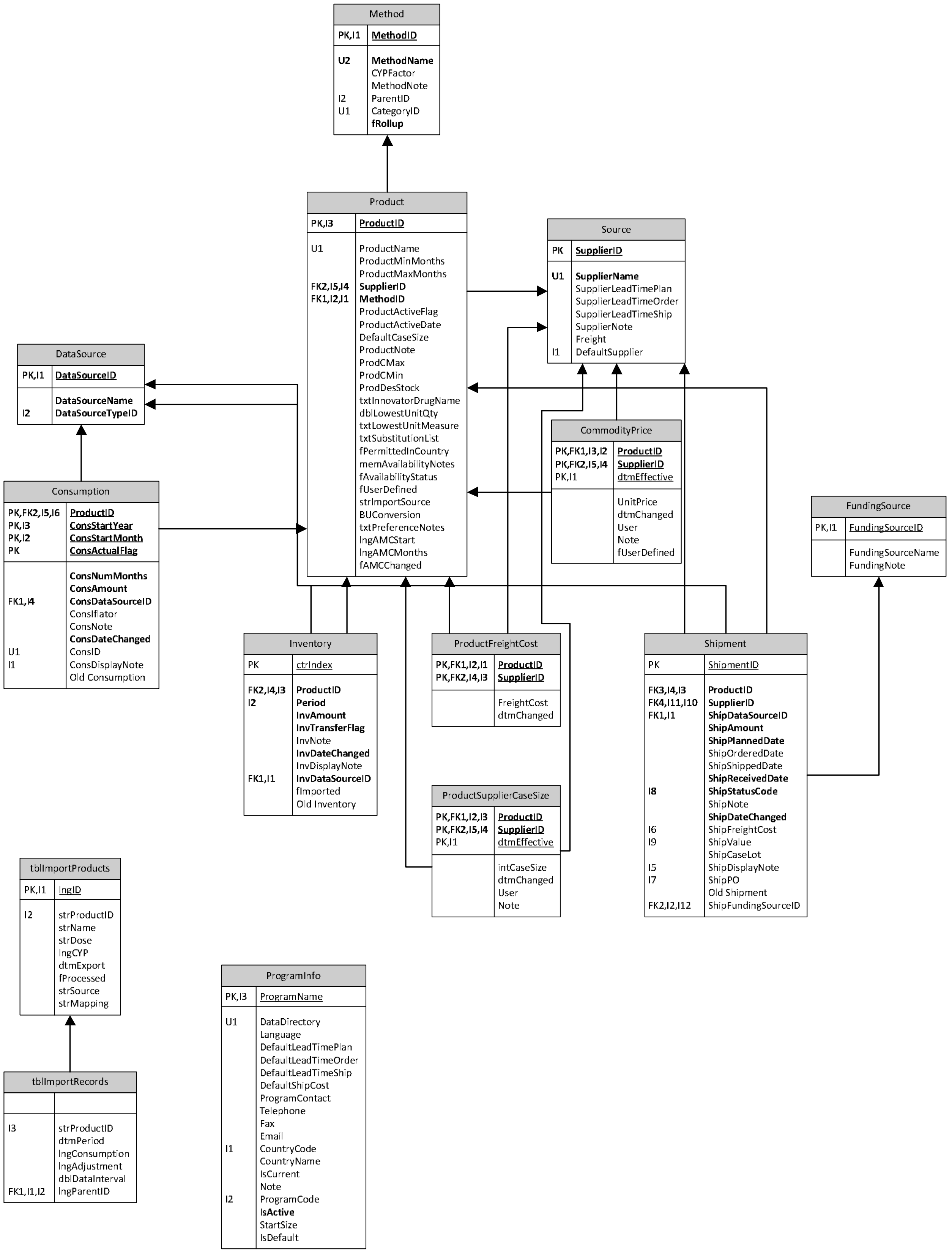


Table and Column Descriptions

Table 15 - List of Tables

TableName	FieldName	DataType	FieldSize	Position	Primary Index	Secondary Index
Action	ActionCode	Text	10	1	Yes	Yes (Duplicates OK)
	ActionName	Text	75	2	No	No
CommodityCost_Temp	ShipmentID	AutoNumber, Long Integer, Increment	4	1	No	No
	ProductID	Text	10	2	No	No
	SupplierID	Text	10	3	No	No
	ShipDataSourceID	Text	10	4	No	No
	ShipAmount	Number, Double	8	5	No	No
	ShipPlannedDate	Date/Time	8	6	No	No
	ShipOrderedDate	Date/Time	8	7	No	No
	ShipShippedDate	Date/Time	8	8	No	No
	ShipReceivedDate	Date/Time	8	9	No	No
	ShipStatusCode	Text	1	10	No	No
	ShipNote	Memo		11	No	No
	ShipDateChanged	Date/Time	8	12	No	No
	ShipFreightCost	Number, Single	4	13	No	No
	ShipValue	Number, Double	8	14	No	No
	ShipCaseLot	Number, Long Integer	4	15	No	No
	ShipDisplayNote	Yes/No	1	16	No	No
	ShipPO	Text	50	17	No	No
	Old Shipment	Number, Double	8	18	No	No
	ShipFundingSourceID	Text	10	19	No	No
	UnitPrice	Text	255	20	No	No
	Value	Number, Double	8	21	No	No

TableName	FieldName	DataType	FieldSize	Position	Primary Index	Secondary Index
	Freight	Number, Double	8	22	No	No
	Cost	Number, Double	8	23	No	No
	ProductName	Text	50	24	No	No
	MethodName	Text	100	25	No	No
	StatusName	Text	50	26	No	No
	SortID	Text	255	27	No	No
CommodityPrice	ProductID	Text	10	1	Yes	Yes (Duplicates OK)
	SupplierID	Text	10	2	Yes	Yes (Duplicates OK)
	dtmEffective	Date/Time	8	3	Yes	Yes (Duplicates OK)
	UnitPrice	Number, Single	4	4	No	No
	dtmChanged	Date/Time	8	5	No	No
	User	Text	35	6	No	No
	Note	Text	255	7	No	No
	fUserDefined	Yes/No	1	8	No	No
Consumption	ProductID	Text	10	1	Yes	Yes (Duplicates OK)
	ConsStartYear	Number, Integer	2	2	Yes	Yes (Duplicates OK)
	ConsStartMonth	Number, Integer	2	3	Yes	Yes (Duplicates OK)
	ConsActualFlag	Yes/No	1	4	Yes	No
	ConsNumMonths	Number, Byte	1	5	No	No
	ConsAmount	Number, Double	8	6	No	No
	ConsDataSourceID	Text	10	7	No	Yes (Duplicates OK)
	ConsIflator	Number, Single	4	8	No	No

TableName	FieldName	DataType	FieldSize	Position	Primary Index	Secondary Index
	ConsNote	Memo		9	No	No
	ConsDateChanged	Date/Time	8	10	No	No
	ConsID	AutoNumber, Long Integer, Increment	4	11	No	Yes (No Duplicates)
	ConsDisplayNote	Yes/No	1	12	No	Yes (Duplicates OK)
	Old Consumption	Number, Double	8	13	No	No
Countries	Code	Text	2	1	Yes	Yes (Duplicates OK)
	Country	Text	50	2	No	No
DataSource	DataSourceID	Text	10	1	Yes	Yes (Duplicates OK)
	DataSourceName	Text	50	2	No	No
	DataSourceTypeID	Text	10	3	No	Yes (Duplicates OK)
DataSourceType	DataSourceTypeID	Text	10	1	Yes	Yes (No Duplicates)
	DataSourceTypeName	Text	50	2	No	Yes (No Duplicates)
FundingSource	FundingSourceID	Text	10	1	Yes	Yes (Duplicates OK)
	FundingSourceName	Text	50	2	No	No
	FundingNote	Memo		3	No	No
Inventory	ProductID	Text	10	1	No	Yes (Duplicates OK)
	Period	Date/Time	8	2	No	Yes (Duplicates OK)
	InvAmount	Number, Double	8	3	No	No
	InvTransferFlag	Yes/No	1	4	No	No
	InvNote	Memo		5	No	No

TableName	FieldName	DataType	FieldSize	Position	Primary Index	Secondary Index
	InvDateChanged	Date/Time	8	6	No	No
	ctrIndex	AutoNumber, Long Integer, Increment	4	7	Yes	Yes (No Duplicates)
	InvDisplayNote	Yes/No	1	8	No	No
	InvDataSourceID	Text	10	9	No	Yes (Duplicates OK)
	fImported	Yes/No	1	10	No	No
	Old Inventory	Number, Double	8	11	No	No
Languages	LangTextID	Text	3	1	Yes	Yes (Duplicates OK)
	LangIntegerID	Number, Integer	2	2	No	Yes (No Duplicates)
	Language01	Text	25	3	No	No
	Language02	Text	25	4	No	No
	Language03	Text	25	5	No	No
	Language04	Text	25	6	No	No
	Language05	Text	25	7	No	No
	NativeText	Text	50	8	No	No
Method	MethodID	Text	15	1	Yes	Yes (Duplicates OK)
	MethodName	Text	100	2	No	Yes (No Duplicates)
	CYPFactor	Number, Single	4	3	No	No
	MethodNote	Memo		4	No	No
	ParentID	Number, Long Integer	4	5	No	Yes (Duplicates OK)
	CategoryID	AutoNumber, Long Integer, Increment	4	6	No	Yes (No Duplicates)
	fRollup	Yes/No	1	7	No	No

TableName	FieldName	DataType	FieldSize	Position	Primary Index	Secondary Index
MonthlyAction	ProductID	Text	10	1	No	Yes (Duplicates OK)
	ActionDate	Date/Time	8	2	No	Yes (Duplicates OK)
	ActionCode	Text	10	3	No	Yes (Duplicates OK)
	StatusCode	Text	10	4	No	Yes (Duplicates OK)
	SupplierID	Text	10	5	No	Yes (Duplicates OK)
	Amount	Number, Long Integer	4	6	No	No
	ShipReceivedDate	Date/Time	8	7	No	No
	ShipShippedDate	Date/Time	8	8	No	No
	ShipOrderedDate	Date/Time	8	9	No	No
	ShipPlannedDate	Date/Time	8	10	No	No
	ShipmentID	Number, Long Integer	4	11	No	Yes (Duplicates OK)
MonthlyConsumption	ProductID	Text	10	1	Yes	Yes (Duplicates OK)
	ConsYear	Number, Integer	2	2	Yes	Yes (Duplicates OK)
	ConsMonth	Number, Byte	1	3	Yes	Yes (Duplicates OK)
	ConsAmount	Number, Long Integer	4	4	No	No
	ConsActualFlag	Yes/No	1	5	No	No
	ConsDataSourceID	Text	10	6	No	Yes (Duplicates OK)
	ConsPreData	Yes/No	1	7	No	Yes (Duplicates OK)
	ConsBU	Number, Long Integer	4	8	No	No

TableName	FieldName	DataType	FieldSize	Position	Primary Index	Secondary Index
MonthlyProblem	ProductID	Text	10	1	No	Yes (Duplicates OK)
	ActionDate	Date/Time	8	2	No	Yes (Duplicates OK)
	ProblemCode	Text	10	3	No	Yes (Duplicates OK)
	StatusCode	Text	10	4	No	Yes (Duplicates OK)
	SupplierID	Text	10	5	No	Yes (Duplicates OK)
	Amount	Number, Long Integer	4	6	No	No
	ShipReceivedDate	Date/Time	8	7	No	No
	ShipShippedDate	Date/Time	8	8	No	No
	ShipOrderedDate	Date/Time	8	9	No	No
	ShipPlannedDate	Date/Time	8	10	No	No
	ShipmentID	Text	10	11	No	Yes (Duplicates OK)
MonthlyShipment	ShipmentID	Number, Long Integer	4	1	Yes	No
	ProductID	Text	10	2	No	Yes
	SupplierID	Text	10	3	No	Yes (Duplicates OK)
	ShipAmount	Number, Double	8	4	No	No
	ShipStatusCode	Text	1	5	No	Yes (Duplicates OK)
	ShipPlannedDate	Date/Time	8	6	No	No
	ShipOrderedDate	Date/Time	8	7	No	No
	ShipShippedDate	Date/Time	8	8	No	No
	ShipReceivedDate	Date/Time	8	9	No	Yes
	ShipDataSourceID	Text	10	10	No	Yes (Duplicates OK)

TableName	FieldName	DataType	FieldSize	Position	Primary Index	Secondary Index
	ShipBU	Number, Double	8	11	No	No
	ShipFundingSourceID	Text	50	12	No	Yes (Duplicates OK)
MonthlyStock	ProductID	Text	10	1	Yes	Yes (Duplicates OK)
	StockYear	Number, Integer	2	2	Yes	Yes (Duplicates OK)
	StockMonth	Number, Byte	1	3	Yes	Yes (Duplicates OK)
	ShipInMonth	Number, Integer	2	4	Yes	No
	dtmRecord	Date/Time	8	5	No	No
	StockShipAmount	Number, Long Integer	4	6	No	No
	StockShipStatus	Text	1	7	No	No
	StockShipSupplier	Text	10	8	No	No
	StockConsAmount	Number, Long Integer	4	9	No	No
	StockAdjustAmount	Number, Long Integer	4	10	No	No
	StockInventoryAutoAdjust	Number, Long Integer	4	11	No	No
	StockShipReceiveAmount	Number, Long Integer	4	12	No	No
	StockBoMAmount	Number, Long Integer	4	13	No	No
	StockAMCAmount	Number, Long Integer	4	14	No	No
	StockShipPlanFlag	Yes/No	1	15	No	No
	StockShipOrderFlag	Yes/No	1	16	No	No
	StockShipShipFlag	Yes/No	1	17	No	No
	ConsActualFlag	Yes/No	1	18	No	No
	StockActualFlag	Yes/No	1	19	No	No
	ShipinQTR	Number, Integer	2	20	No	No
	MethodShipinMon	Number, Integer	2	21	No	No
	MethodShipinQTR	Number, Integer	2	22	No	No

TableName	FieldName	DataType	FieldSize	Position	Primary Index	Secondary Index
	ProductBU	Number, Long Integer	4	23	No	No
	StockInAmount	Number, Long Integer	4	24	No	No
	StockOutAmount	Number, Long Integer	4	25	No	No
	StockShipFundingSourceID	Text	50	26	No	Yes (Duplicates OK)
Period	Period_Name	Text	50	1	No	Yes (No Duplicates)
	Period_Code	Number, Integer	2	2	Yes	No
	Month	Yes/No	1	3	No	No
	Beg_Month	Text	12	4	No	No
	End_Month	Text	12	5	No	No
	intMinMonths	Number, Long Integer	4	6	No	No
	Period_Name_Arabic	Text	12	7	No	Yes (No Duplicates)
	Beg_Month_Arabic	Text	12	8	No	No
	End_Month_Arabic	Text	12	9	No	No
	Period_Name_Sp	Text	50	10	No	Yes (No Duplicates)
	Period_Name_Port	Text	50	11	No	Yes (No Duplicates)
	Period_Name_Fra	Text	50	12	No	Yes (No Duplicates)
	Period_Name_Eng	Text	50	13	No	Yes (No Duplicates)
Problem	ProblemCode	Text	10	1	Yes	Yes (No Duplicates)
	ProblemName	Text	50	2	No	No
Product	ProductID	Text	10	1	Yes	Yes (Duplicates OK)
	ProductName	Text	50	2	No	Yes (No

TableName	FieldName	DataType	FieldSize	Position	Primary Index	Secondary Index
						Duplicates)
	ProductMinMonths	Number, Byte	1	3	No	No
	ProductMaxMonths	Number, Byte	1	4	No	No
	SupplierID	Text	10	5	No	Yes (Duplicates OK)
	MethodID	Text	15	6	No	Yes (Duplicates OK)
	ProductActiveFlag	Yes/No	1	7	No	No
	ProductActiveDate	Date/Time	8	8	No	No
	DefaultCaseSize	Number, Long Integer	4	9	No	No
	ProductNote	Memo		10	No	No
	ProdCMax	Number, Byte	1	11	No	No
	ProdCMin	Number, Byte	1	12	No	No
	ProdDesStock	Number, Byte	1	13	No	No
	txtInnovatorDrugName	Text	50	14	No	No
	dblLowestUnitQty	Number, Double	8	15	No	No
	txtLowestUnitMeasure	Text	25	16	No	No
	txtSubstitutionList	Text	255	17	No	No
	fPermittedInCountry	Yes/No	1	18	No	No
	memAvailabilityNotes	Memo		19	No	No
	fAvailabilityStatus	Yes/No	1	20	No	No
	fUserDefined	Yes/No	1	21	No	No
	strImportSource	Text	10	22	No	No
	BUConversion	Number, Long Integer	4	23	No	No
	txtPreferenceNotes	Text	255	24	No	No
	lngAMCStart	Number, Long Integer	4	25	No	No
	lngAMCMonths	Number, Long Integer	4	26	No	No
	fAMCChanged	Yes/No	1	27	No	No

TableName	FieldName	DataType	FieldSize	Position	Primary Index	Secondary Index
ProductFreightCost	ProductID	Text	10	1	Yes	Yes (Duplicates OK)
	SupplierID	Text	10	2	Yes	Yes (Duplicates OK)
	FreightCost	Number, Single	4	3	No	No
	dtmChanged	Date/Time	8	4	No	No
ProductSupplierCaseSize	ProductID	Text	10	1	Yes	Yes (Duplicates OK)
	SupplierID	Text	10	2	Yes	Yes (Duplicates OK)
	dtmEffective	Date/Time	8	3	Yes	Yes (Duplicates OK)
	intCaseSize	Number, Long Integer	4	4	No	No
	dtmChanged	Date/Time	8	5	No	No
	User	Text	35	6	No	No
	Note	Text	255	7	No	No
Program	ProgramName	Text	50	1	Yes	Yes (Duplicates OK)
	DataDirectory	Text	250	2	No	Yes (No Duplicates)
	Language	Text	3	3	No	No
	DefaultLeadTimePlan	Number, Single	4	4	No	No
	DefaultLeadTimeOrder	Number, Single	4	5	No	No
	DefaultLeadTimeShip	Number, Single	4	6	No	No
	DefaultShipCost	Number, Single	4	7	No	No
	ProgramContact	Text	50	8	No	No
	Telephone	Text	50	9	No	No
	Fax	Text	50	10	No	No
	Email	Text	50	11	No	No

TableName	FieldName	DataType	FieldSize	Position	Primary Index	Secondary Index
	CountryCode	Text	2	12	No	Yes (Duplicates OK)
	CountryName	Text	50	13	No	No
	IsCurrent	Yes/No	1	14	No	No
	Note	Memo		15	No	No
	ProgramCode	Text	12	16	No	Yes (No Duplicates)
	IsActive	Yes/No	1	17	No	No
	StartSize	Text	50	18	No	No
	IsDefault	Yes/No	1	19	No	No
	fStartupFile	Yes/No	1	20	No	No
Shipment	ShipmentID	AutoNumber, Long Integer, Increment	4	1	Yes	Yes (No Duplicates)
	ProductID	Text	10	2	No	Yes (Duplicates OK)
	SupplierID	Text	10	3	No	Yes (Duplicates OK)
	ShipDataSourceID	Text	10	4	No	Yes (Duplicates OK)
	ShipAmount	Number, Double	8	5	No	No
	ShipPlannedDate	Date/Time	8	6	No	No
	ShipOrderedDate	Date/Time	8	7	No	No
	ShipShippedDate	Date/Time	8	8	No	No
	ShipReceivedDate	Date/Time	8	9	No	No
	ShipStatusCode	Text	1	10	No	Yes (Duplicates OK)
	ShipNote	Memo		11	No	No
	ShipDateChanged	Date/Time	8	12	No	No
	ShipFreightCost	Number, Single	4	13	No	Yes (Duplicates

TableName	FieldName	DataType	FieldSize	Position	Primary Index	Secondary Index
						OK)
	ShipValue	Number, Double	8	14	No	Yes (Duplicates OK)
	ShipCaseLot	Number, Long Integer	4	15	No	No
	ShipDisplayNote	Yes/No	1	16	No	Yes (Duplicates OK)
	ShipPO	Text	50	17	No	Yes (Duplicates OK)
	Old Shipment	Number, Double	8	18	No	No
	ShipFundingSourceID	Text	10	19	No	Yes (Duplicates OK)
ShipSchedule	ProductID	Text	10	1	No	Yes (Duplicates OK)
	ActionDate	Date/Time	8	2	No	Yes (Duplicates OK)
	ActionCode	Text	10	3	No	Yes (Duplicates OK)
	StatusCode	Text	10	4	No	Yes (Duplicates OK)
	SupplierID	Text	10	5	No	Yes (Duplicates OK)
	Amount	Number, Long Integer	4	6	No	No
	ShipReceivedDate	Date/Time	8	7	No	No
	ShipShippedDate	Date/Time	8	8	No	No
	ShipOrderedDate	Date/Time	8	9	No	No
	ShipPlannedDate	Date/Time	8	10	No	No
	ShipmentID	Number, Long Integer	4	11	No	Yes (Duplicates OK)
Source	SupplierID	Text	10	1	Yes	Yes (No Duplicates)

TableName	FieldName	DataType	FieldSize	Position	Primary Index	Secondary Index
	SupplierName	Text	50	2	No	Yes (No Duplicates)
	SupplierLeadTimePlan	Number, Single	4	3	No	No
	SupplierLeadTimeOrder	Number, Single	4	4	No	No
	SupplierLeadTimeShip	Number, Single	4	5	No	No
	SupplierNote	Memo		6	No	No
	Freight	Number, Single	4	7	No	No
	DefaultSupplier	Yes/No	1	8	No	Yes (Duplicates OK)
Status	StatusCode	Text	1	1	Yes	Yes (Duplicates OK)
	StatusName	Text	50	2	No	Yes (No Duplicates)
	StatusOrder	Number, Integer	2	3	No	No
	CableOrder	Number, Integer	2	4	No	No
Switchboard Items	SwitchboardID	Number, Long Integer	4	1	Yes	No
	ItemNumber	Number, Integer	2	2	Yes	No
	ItemText	Text	255	3	No	No
	Command	Number, Integer	2	4	No	No
	Argument	Text	50	5	No	No
	TransCode	Text	50	6	No	Yes (Duplicates OK)
tblComments	cmt_strProductID	Text	10	1	No	Yes (Duplicates OK)
	cmt_dtmPeriod	Date/Time	8	2	No	No
	cmt_strSource	Text	50	3	No	No
	cmt_memNote	Memo		4	No	No
tblDisplayBy	DisplayByID	Number, Long Integer	4	1	No	Yes (Duplicates OK)

TableName	FieldName	DataType	FieldSize	Position	Primary Index	Secondary Index
	DisplayByName	Text	50	2	No	Yes (No Duplicates)
tblEstimates	Month	Date/Time	8	1	Yes	No
	intOrder	Number, Integer	2	2	No	Yes (Duplicates OK)
	Estimated	Number, Double	8	3	No	No
	hasActual	Yes/No	1	4	No	No
	Actual	Number, Long Integer	4	5	No	No
	Forecast	Number, Long Integer	4	6	No	No
	LinEst	Number, Double	8	7	No	No
	LinMin	Number, Double	8	8	No	No
	LinMax	Number, Double	8	9	No	No
tblEstimatesForecast	Month	Date/Time	8	1	Yes	No
	intOrder	Number, Integer	2	2	No	Yes (Duplicates OK)
	Estimated	Number, Double	8	3	No	No
	hasActual	Yes/No	1	4	No	No
	Actual	Number, Long Integer	4	5	No	No
	Forecast	Number, Long Integer	4	6	No	No
	LinEst	Number, Double	8	7	No	No
	LinMin	Number, Double	8	8	No	No
	LinMax	Number, Double	8	9	No	No
	ConsID	Number, Long Integer	4	10	No	No
	fForecast	Yes/No	1	11	No	No
tblHelpContextID	FormName	Text	255	1	Yes	No
	IsForm	Yes/No	1	2	Yes	No
	Index	Number, Byte	1	3	Yes	No
	EnglishContextID	Number, Long Integer	4	4	No	Yes (Duplicates

TableName	FieldName	DataType	FieldSize	Position	Primary Index	Secondary Index
						OK)
	FrenchContextID	Number, Long Integer	4	5	No	Yes (Duplicates OK)
	SpanishContextID	Number, Long Integer	4	6	No	Yes (Duplicates OK)
	Comment	Text	255	7	No	No
	ArabicContextID	Number, Long Integer	4	8	No	Yes (Duplicates OK)
	PortugueseContextID	Number, Long Integer	4	9	No	Yes (Duplicates OK)
tblImportProducts	strProductID	Text	20	1	No	Yes (Duplicates OK)
	strName	Text	100	2	No	No
	strDose	Text	100	3	No	No
	lngCYP	Number, Double	8	4	No	No
	dtmExport	Date/Time	8	5	No	No
	fProcessed	Yes/No	1	6	No	No
	lngID	AutoNumber, Long Integer, Increment	4	7	Yes	Yes (Duplicates OK)
	strSource	Text	255	8	No	No
	strMapping	Text	50	9	No	No
tblImportProductsSCMS	strProductID	Text	20	1	No	Yes (Duplicates OK)
	strName	Text	255	2	No	No
	strDose	Text	100	3	No	No
	lngCYP	Number, Double	8	4	No	No
	dtmExport	Date/Time	8	5	No	No
	fProcessed	Yes/No	1	6	No	No
	lngID	AutoNumber, Long Integer, Increment	4	7	Yes	Yes (Duplicates OK)

TableName	FieldName	DataType	FieldSize	Position	Primary Index	Secondary Index
	strSource	Text	255	8	No	No
	strMapping	Text	255	9	No	No
	strMappingFull	Text	255	10	No	No
	strPLMapping	Text	255	11	No	No
	strPLMappingFull	Text	255	12	No	No
	fSelect	Yes/No	1	13	No	No
	fLocked	Yes/No	1	14	No	No
	fNoBlanks	Yes/No	1	15	No	No
	fUserDefined	Yes/No	1	16	No	No
	strProductGroup	Text	50	17	No	No
	strInnovatorName	Text	50	18	No	No
	dblLowestUnitQty	Number, Long Integer	4	19	No	No
	strLowestUnitMeasure	Text	50	20	No	No
	intQuantificationFactor	Number, Double	8	21	No	No
	strSourceName	Text	50	22	No	No
	strSystemName	Text	50	23	No	No
	strShortName	Text	50	24	No	No
	fPLLocked	Yes/No	1	25	No	No
tblImportRecords	strProductID	Text	50	1	No	Yes (Duplicates OK)
	dtmPeriod	Date/Time	8	2	No	No
	lngConsumption	Number, Long Integer	4	3	No	No
	lngAdjustment	Number, Long Integer	4	4	No	No
	dblDataInterval	Number, Double	8	5	No	No
	lngParentID	Number, Long Integer	4	6	No	Yes (Duplicates OK)
tblImportRecordsSCMS	strProductID	Text	50	1	No	Yes (Duplicates OK)

TableName	FieldName	DataType	FieldSize	Position	Primary Index	Secondary Index
	dtmPeriod	Date/Time	8	2	No	No
	lngConsumption	Number, Double	8	3	No	No
	lngAdjustment	Number, Long Integer	4	4	No	No
	dblDataInterval	Number, Double	8	5	No	No
	lngParentID	Number, Long Integer	4	6	No	Yes (Duplicates OK)
tblInterpolate	Month	Date/Time	8	1	Yes	No
	intOrder	Number, Integer	2	2	No	Yes (Duplicates OK)
	intercept	Number, Double	8	3	No	No
	slope	Number, Double	8	4	No	No
	Estimated	Number, Double	8	5	No	No
	fForecast	Yes/No	1	6	No	No
	ConsID	Number, Long Integer	4	7	No	No
tblMakeGraph_Temp	ProductID	Text	10	1	No	Yes (Duplicates OK)
	Date	Date/Time	8	2	No	No
	Actual	Number, Long Integer	4	3	No	No
	Forecast	Number, Long Integer	4	4	No	No
	ConsAmount	Number, Long Integer	4	5	No	No
	ConsActualFlag	Yes/No	1	6	No	No
tblStatusbartext	FormName	Text	255	1	Yes	No
	ControlName	Text	255	2	Yes	No
	Index	Number, Byte	1	3	Yes	No
	EnglishStatus	Text	255	4	No	No
	FrenchStatus	Text	255	5	No	No
	SpanishStatus	Text	255	6	No	No
	Comment	Text	255	7	No	No

TableName	FieldName	DataType	FieldSize	Position	Primary Index	Secondary Index
	property	Number, Long Integer	4	8	No	No
	ID	Text	50	9	No	Yes (Duplicates OK)
tblStyle	lngType	AutoNumber, Long Integer, Increment	4	1	Yes	No
	txtTypeName	Text	50	2	No	No
	txtFont	Text	50	3	No	No
	lngFontSize	Number, Long Integer	4	4	No	No
	txtForeColor	Text	50	5	No	No
	txtFontweight	Text	50	6	No	No
	fUnderline	Yes/No	1	7	No	No
	lngSpecialEffect	Number, Long Integer	4	8	No	No
	lngBorderStyle	Number, Long Integer	4	9	No	No
	txtBackColor	Text	50	10	No	No
tblSysParameters	strParmName	Text	50	1	Yes	No
	strParmValue	Text	50	2	No	No
	strParmDesc	Text	255	3	No	No
tblTempCPT	r_code	Text	12	1	Yes	Yes (Duplicates OK)
	p_code	Text	10	2	Yes	Yes (Duplicates OK)
	tb_yr	Number, Integer	2	3	No	No
	tb_Prep	Date/Time	8	4	No	No
	tb_who	Text	50	5	No	No
	tb_min_lvl1	Number, Integer	2	6	No	No
	tb_min_lvl2	Number, Integer	2	7	No	No
	tb_m_eoy1	Number, Integer	2	8	No	No
	tb_m_eoy2	Number, Integer	2	9	No	No
	tb_begQty	Number, Double	8	10	No	No

TableName	FieldName	DataType	FieldSize	Position	Primary Index	Secondary Index
	tb_est1	Number, Single	4	11	No	No
	tb_est2	Number, Single	4	12	No	No
	tb_est3	Number, Single	4	13	No	No
	tb_est4	Number, Single	4	14	No	No
	tb_est5	Number, Single	4	15	No	No
	tb_lost1	Number, Single	4	16	No	No
	tb_lost2	Number, Single	4	17	No	No
	tb_lost3	Number, Single	4	18	No	No
	tb_lost4	Number, Single	4	19	No	No
	tb_lost5	Number, Single	4	20	No	No
	tb_AidIn1	Number, Single	4	21	No	No
	tb_AidIn2	Number, Single	4	22	No	No
	tb_AidIn3	Number, Single	4	23	No	No
	tb_AidDue2	Number, Single	4	24	No	No
	tb_AidDue3	Number, Single	4	25	No	No
	tb_AidDue4	Number, Single	4	26	No	No
	tb_AidDue5	Number, Single	4	27	No	No
	tb_des3	Number, Single	4	28	No	No
	tb_des4	Number, Single	4	29	No	No
	tb_des5	Number, Single	4	30	No	No
	tb_Ord3	Number, Single	4	31	No	No
	tb_Ord4	Number, Single	4	32	No	No
	tb_Ord5	Number, Single	4	33	No	No
	tb_TransIn1	Number, Single	4	34	No	No
	tb_TransIn2	Number, Single	4	35	No	No
	tb_TransIn3	Number, Single	4	36	No	No
	tb_TransIn4	Number, Single	4	37	No	No

TableName	FieldName	DataType	FieldSize	Position	Primary Index	Secondary Index
	tb_TransIn5	Number, Single	4	38	No	No
	tb_TransOut1	Number, Single	4	39	No	No
	tb_TransOut2	Number, Single	4	40	No	No
	tb_TransOut3	Number, Single	4	41	No	No
	tb_TransOut4	Number, Single	4	42	No	No
	tb_TransOut5	Number, Single	4	43	No	No
	tb_Max_lvl1	Number, Integer	2	44	No	No
	tb_Max_lvl2	Number, Integer	2	45	No	No
tblTempCPTProducts	ProductID	Text	10	1	Yes	No
tblTempCPTShipments	ShipmentID	Number, Long Integer	4	1	Yes	No
	ProductID	Text	10	2	No	Yes
	SupplierID	Text	10	3	No	Yes (Duplicates OK)
	ShipAmount	Number, Double	8	4	No	No
	ShipStatusCode	Text	1	5	No	Yes (Duplicates OK)
	ShipPlannedDate	Date/Time	8	6	No	No
	ShipOrderedDate	Date/Time	8	7	No	No
	ShipShippedDate	Date/Time	8	8	No	No
	ShipReceivedDate	Date/Time	8	9	No	Yes
	ShipDataSourceID	Text	10	10	No	Yes (Duplicates OK)
	ShipBU	Number, Double	8	11	No	No
tblTempImportShipments	Recipient	Text	255	1	No	Yes (Duplicates OK)
	ProductID	Text	255	2	No	Yes (Duplicates OK)
	ReceiptDate	Date/Time	8	3	No	No
	Quantity	Number, Double	8	4	No	No

TableName	FieldName	DataType	FieldSize	Position	Primary Index	Secondary Index
	StatusCode	Text	255	5	No	Yes (Duplicates OK)
	newvernID	Text	255	6	No	Yes (Duplicates OK)
	Comment	Text	255	7	No	No
tblTempNewvernShipments	Recipient	Text	255	1	No	Yes (Duplicates OK)
	ProductID	Text	255	2	No	Yes (Duplicates OK)
	TB_YR	Number, Integer	2	3	No	No
	ReceiptDate	Date/Time	8	4	No	No
	Quantity	Number, Double	8	5	No	No
	StatusCode	Text	255	6	No	Yes (Duplicates OK)
	NewvernID	Text	255	7	No	Yes (Duplicates OK)
	Comment	Text	255	8	No	No
tempForeCons	ProductID	Text	10	1	No	No
	ConsYear	Number, Integer	2	2	Yes	No
	ConsMonth	Number, Integer	2	3	Yes	No
	ConsAmount	Number, Long Integer	4	4	No	No
	ConsDataSourceID	Text	10	5	No	No
tempXtabStockStatusMatrix	KeyID	Text	10	1	No	No
	KeyName	Text	50	2	No	No
	Unit	Text	25	3	No	No
	StockMin	Number, Byte	1	4	No	No
	StockMax	Number, Byte	1	5	No	No
	MOS	Number, Double	8	6	No	No
	StockDate	Date/Time	8	7	No	No

TableName	FieldName	DataType	FieldSize	Position	Primary Index	Secondary Index
tlkReportExport	ProductID	Text	10	1	No	No
	txtMonYear	Date/Time	8	2	No	No
	txtStockBoMAmount	Number, Long Integer	4	3	No	No
	StockShipAmount	Number, Long Integer	4	4	No	No
	txtStockShipStatus	Text	255	5	No	No
	txtStockShipSupplier	Text	255	6	No	No
	txtActCons	Number, Long Integer	4	7	No	No
	Actual	Text	255	8	No	No
	TotalAdjustAmount	Number, Long Integer	4	9	No	No
	txtStockinMonths	Number, Double	8	10	No	No
	txttoMax	Number, Double	8	11	No	No
	txtShortSurp	Number, Double	8	12	No	No
	txtEoMAmount	Number, Long Integer	4	13	No	No
tlkTranslationText	DocumentType	Number, Long Integer	4	1	Yes	No
	FormName	Text	255	2	Yes	No
	ControlName	Text	255	3	Yes	No
	Property	Number, Long Integer	4	4	Yes	No
	Index	Number, Byte	1	5	Yes	No
	TranslationID	Text	255	6	No	Yes (Duplicates OK)
	StyleID	Number, Long Integer	4	7	No	Yes (Duplicates OK)
	txtJustification	Number, Long Integer	4	8	No	No
	FrenchStatus	Text	255	9	No	No
	SpanishStatus	Text	255	10	No	No
	EnglishStatus	Text	255	11	No	No
	NewRecord	Yes/No	1	12	No	No
	ArabicStatus	Text	255	13	No	No

TableName	FieldName	DataType	FieldSize	Position	Primary Index	Secondary Index
	bytReadingOrderA	Number, Byte	1	14	No	No
	PortugueseStatus	Text	255	15	No	No
	comment	Text	50	16	No	No
tlkTreeview	lngID	AutoNumber, Long Integer, Increment	4	1	Yes	Yes (Duplicates OK)
	dblParent	Number, Double	8	2	No	Yes
	dblsortorder	Number, Double	8	3	No	Yes
	txtName	Text	50	4	No	No
	txtformname	Text	50	5	No	No
	txtSpanish	Text	50	6	No	No
	txtFrench	Text	50	7	No	No
	NewRecord	Yes/No	1	8	No	No
	TranslationID	Text	255	9	No	Yes (Duplicates OK)
	txtArabic	Text	50	10	No	No
	txtPortuguese	Text	50	11	No	No
tlkUpgrade	id	AutoNumber, Long Integer, Increment	4	1	Yes	Yes (Duplicates OK)
	Type	Text	50	2	No	No
	strSQL1	Memo		3	No	No
	strSQL2	Memo		4	No	No
	strSQL3	Memo		5	No	No
	strTable	Text	50	6	No	No
	version	Number, Long Integer	4	7	No	No
	lngOrder	Number, Long Integer	4	8	No	No
tlkYears	lngYear	Number, Long Integer	4	1	No	No
tmpCaseSizes	ProductID	Text	10	1	No	Yes (Duplicates OK)

TableName	FieldName	DataType	FieldSize	Position	Primary Index	Secondary Index
	SupplierID	Text	10	2	No	Yes (Duplicates OK)
	dtmEffective	Date/Time	8	3	No	Yes (Duplicates OK)
	intCaseSize	Number, Long Integer	4	4	No	No
	dtmChanged	Date/Time	8	5	No	No
	User	Text	35	6	No	No
	Note	Text	255	7	No	No
tmpCategory	MethodID	Text	10	1	No	Yes (Duplicates OK)
	MethodName	Text	50	2	No	No
	CYPFactor	Number, Single	4	3	No	No
	MethodNote	Memo		4	No	No
	fRollup	Yes/No	1	5	No	No
	ParentID	Text	255	6	No	Yes (Duplicates OK)
tmpCategoryProduct	MethodID	Text	10	1	No	Yes (Duplicates OK)
	ProductID	Text	10	2	No	Yes (Duplicates OK)
tmpCloneConsumption	ProductID	Text	10	1	No	Yes (Duplicates OK)
	ConsStartYear	Number, Integer	2	2	No	Yes (Duplicates OK)
	ConsStartMonth	Number, Integer	2	3	No	Yes (Duplicates OK)
	ConsActualFlag	Yes/No	1	4	No	No
	ConsNumMonths	Number, Byte	1	5	No	No
	ConsAmount	Number, Double	8	6	No	No
	ConsDataSourceID	Text	10	7	No	Yes (Duplicates

TableName	FieldName	DataType	FieldSize	Position	Primary Index	Secondary Index
						OK)
	ConsIflator	Number, Single	4	8	No	No
	ConsNote	Memo		9	No	No
	ConsDateChanged	Date/Time	8	10	No	No
	ConsDisplayNote	Yes/No	1	11	No	Yes (Duplicates OK)
	ConsAmountBU	Number, Double	8	12	No	No
	ConsClonedAmountBU	Number, Double	8	13	No	No
	ConsClonedProductID	Text	10	14	No	Yes (Duplicates OK)
	ConsfDataAccepted	Yes/No	1	15	No	No
	ConsfConflict	Yes/No	1	16	No	No
tmpCloneProducts	ProductID	Text	10	1	No	Yes (Duplicates OK)
	ProductName	Text	50	2	No	No
	dblLowestUnitQty	Number, Double	8	3	No	No
	fSelected	Number, Long Integer	4	4	No	No
	dblFactor	Number, Double	8	5	No	No
tmpConsumption	ProductID	Text	10	1	No	Yes (Duplicates OK)
	ConsStartYear	Number, Integer	2	2	No	Yes (Duplicates OK)
	ConsStartMonth	Number, Integer	2	3	No	Yes (Duplicates OK)
	ConsActualFlag	Yes/No	1	4	No	No
	ConsNumMonths	Number, Byte	1	5	No	No
	ConsAmount	Number, Double	8	6	No	No
	ConsDataSourceID	Text	10	7	No	Yes (Duplicates OK)

TableName	**FieldName**	**DataType**	**FieldSize**	**Position**	**Primary Index**	**Secondary Index**
	ConsIflator	Number, Single	4	8	No	No
	ConsNote	Memo		9	No	No
	ConsDateChanged	Date/Time	8	10	No	No
	ConsDisplayNote	Yes/No	1	11	No	Yes (Duplicates OK)
tmpCosts	ProductID	Text	10	1	No	Yes (Duplicates OK)
	SupplierID	Text	10	2	No	Yes (Duplicates OK)
	dtmEffective	Date/Time	8	3	No	Yes (Duplicates OK)
	UnitPrice	Number, Single	4	4	No	No
	dtmChanged	Date/Time	8	5	No	No
	User	Text	35	6	No	No
	Note	Text	255	7	No	No
tmpDataSources	DataSourceID	Text	10	1	No	Yes (No Duplicates)
	DataSourceName	Text	50	2	No	No
	DataSourceTypeID	Text	10	3	No	Yes (Duplicates OK)
tmpFundingSources	FundingSourceID	Text	10	1	Yes	Yes (Duplicates OK)
	FundingSourceName	Text	50	2	No	No
	FundingNote	Memo		3	No	No
tmpImp_Category	MethodID	Text	50	1	Yes	Yes (Duplicates OK)
	ParentID	Text	50	2	No	Yes (Duplicates OK)
	MethodName	Text	50	3	No	Yes (No Duplicates)

TableName	FieldName	DataType	FieldSize	Position	Primary Index	Secondary Index
	fRollup	Yes/No	1	4	No	No
	id	Number, Long Integer	4	5	No	Yes (No Duplicates)
tmpImp_Country	Code	Text	2	1	Yes	Yes (Duplicates OK)
	Country	Text	50	2	No	No
tmpImp_Price	ProductID	Text	10	1	Yes	Yes (Duplicates OK)
	SupplierID	Text	10	2	Yes	Yes (Duplicates OK)
	dtmEffective	Date/Time	8	3	Yes	Yes (Duplicates OK)
	UnitPrice	Number, Single	4	4	No	No
	dtmChanged	Date/Time	8	5	No	No
tmpImp_Product	ProductID	Text	10	1	Yes	Yes (Duplicates OK)
	ProductName	Text	50	2	No	Yes (No Duplicates)
	SupplierID	Text	10	3	No	Yes (Duplicates OK)
	MethodID	Text	50	4	No	Yes (Duplicates OK)
	DefaultCaseSize	Number, Long Integer	4	5	No	No
	txtInnovatorDrugName	Text	50	6	No	No
	dblLowestUnitQty	Number, Double	8	7	No	No
	txtLowestUnitMeasure	Text	25	8	No	No
	txtSubstitutionList	Text	255	9	No	No
	fPermittedInCountry	Yes/No	1	10	No	No
	fAvailabilityStatus	Yes/No	1	11	No	No
	txtComments	Text	50	12	No	No

TableName	FieldName	DataType	FieldSize	Position	Primary Index	Secondary Index
	fMapping	Yes/No	1	13	No	No
	strMapping	Text	50	14	No	No
	strMappingFull	Text	70	15	No	No
	fLocked	Yes/No	1	16	No	No
	fskipped	Yes/No	1	17	No	No
	txtMethodDisplay	Text	255	18	No	No
	dtmExported	Date/Time	8	19	No	No
	txtSource	Text	255	20	No	No
	fSCMSValidity	Yes/No	1	21	No	No
	fUserDefined	Yes/No	1	22	No	No
	intQuantificationFactor	Number, Long Integer	4	23	No	No
tmpImp_Subcategory	MethodID	Text	10	1	Yes	Yes (Duplicates OK)
	MethodName	Text	50	2	No	Yes (No Duplicates)
	ParentID	Text	50	3	No	Yes (Duplicates OK)
	fRollup	Yes/No	1	4	No	No
tmpImp_Supplier	SupplierID	Text	10	1	Yes	Yes (No Duplicates)
	SupplierName	Text	50	2	No	Yes (No Duplicates)
tmpImportProducts	strProductID	Text	20	1	No	Yes (Duplicates OK)
	strName	Text	100	2	No	No
	strDose	Text	100	3	No	No
	lngCYP	Number, Double	8	4	No	No
	dtmExport	Date/Time	8	5	No	No
	fProcessed	Yes/No	1	6	No	No

TableName	FieldName	DataType	FieldSize	Position	Primary Index	Secondary Index
	lngID	Number, Long Integer	4	7	No	Yes (No Duplicates)
	strSource	Text	255	8	No	No
	strMapping	Text	50	9	No	No
tmpImportProductsSCMS	strProductID	Text	20	1	No	Yes (Duplicates OK)
	strName	Text	100	2	No	No
	strDose	Text	100	3	No	No
	lngCYP	Number, Double	8	4	No	No
	dtmExport	Date/Time	8	5	No	No
	fProcessed	Yes/No	1	6	No	No
	lngID	Number, Long Integer	4	7	No	Yes (No Duplicates)
	strSource	Text	255	8	No	No
	strMapping	Text	50	9	No	No
tmpImportRecords	Product	Text	50	1	No	No
	StartYear	Number, Integer	2	2	No	No
	StartMonth	Number, Integer	2	3	No	No
	dtmPeriod	Date/Time	8	4	No	No
	lngConsumption	Number, Long Integer	4	5	No	No
	lngAdjustment	Number, Long Integer	4	6	No	No
	dblDataInterval	Number, Double	8	7	No	No
	strSource	Text	255	8	No	No
	ProductID	Number, Long Integer	4	9	No	Yes (Duplicates OK)
tmpImportRecordsSCMS	Product	Text	50	1	No	No
	StartYear	Number, Integer	2	2	No	No
	StartMonth	Number, Integer	2	3	No	No

TableName	FieldName	DataType	FieldSize	Position	Primary Index	Secondary Index
	dtmPeriod	Date/Time	8	4	No	No
	lngConsumption	Number, Double	8	5	No	No
	lngAdjustment	Number, Long Integer	4	6	No	No
	dblDataInterval	Number, Double	8	7	No	No
	ProductID	Number, Long Integer	4	8	No	Yes (Duplicates OK)
tmpImportRecordsSHIP	ShipmentID	Number, Long Integer	4	1	No	Yes (Duplicates OK)
	ProductID	Text	10	2	No	Yes (Duplicates OK)
	SupplierID	Text	10	3	No	Yes (Duplicates OK)
	ShipAmount	Number, Double	8	4	No	No
	ShipPlannedDate	Date/Time	8	5	No	No
	ShipReceivedDate	Date/Time	8	6	No	No
	ShipStatusCode	Text	1	7	No	Yes (Duplicates OK)
	ShipNote	Memo		8	No	No
	ShipDateChanged	Date/Time	8	9	No	No
	ShipFreightCost	Number, Double	8	10	No	No
	ShipValue	Number, Double	8	11	No	No
	ShipCaseLot	Number, Long Integer	4	12	No	No
	ShipDisplayNote	Yes/No	1	13	No	No
	ShipPO	Text	50	14	No	No
	strMapping	Text	50	15	No	No
	strMappingFull	Text	70	16	No	No
	fLocked	Yes/No	1	17	No	No
	fDataSourceID	Yes/No	1	18	No	No
	fSupplierID	Yes/No	1	19	No	No

TableName	**FieldName**	**DataType**	**FieldSize**	**Position**	**Primary Index**	**Secondary Index**
	fProductID	Yes/No	1	20	No	No
	ProductName	Text	50	21	No	No
	fMapping	Yes/No	1	22	No	No
	dtmExported	Date/Time	8	23	No	No
	txtSource	Text	255	24	No	No
	ShipDataSourceID	Text	50	25	No	Yes (Duplicates OK)
	ShipFundingSourceID	Text	50	26	No	Yes (Duplicates OK)
	fFundingSourceID	Yes/No	1	27	No	No
	fSplit	Yes/No	1	28	No	No
	lngID	Number, Long Integer	4	29	No	Yes (No Duplicates)
tmpInventory	ProductID	Text	10	1	No	Yes (Duplicates OK)
	Period	Date/Time	8	2	No	Yes (Duplicates OK)
	InvAmount	Number, Double	8	3	No	No
	InvTransferFlag	Yes/No	1	4	No	No
	InvDataSourceID	Text	10	5	No	Yes (Duplicates OK)
	InvNote	Memo		6	No	No
	InvDateChanged	Date/Time	8	7	No	No
	InvDisplayNote	Yes/No	1	8	No	No
tmpProducts	ProductID	Text	10	1	No	Yes (No Duplicates)
	ProductName	Text	50	2	No	Yes (No Duplicates)
	ProductMinMonths	Number, Byte	1	3	No	No

TableName	FieldName	DataType	FieldSize	Position	Primary Index	Secondary Index
	ProductMaxMonths	Number, Byte	1	4	No	No
	SupplierID	Text	10	5	No	Yes (Duplicates OK)
	ProductActiveFlag	Yes/No	1	6	No	No
	ProductActiveDate	Date/Time	8	7	No	No
	DefaultCaseSize	Number, Long Integer	4	8	No	No
	ProductNote	Memo		9	No	No
	ProdCMax	Number, Byte	1	10	No	No
	ProdCMin	Number, Byte	1	11	No	No
	ProdDesStock	Number, Byte	1	12	No	No
	MethodID	Text	10	13	No	Yes (Duplicates OK)
tmpProgram	ProgramName	Text	50	1	No	No
	DataDirectory	Text	250	2	No	No
	Language	Text	3	3	No	No
	DefaultLeadTimePlan	Number, Single	4	4	No	No
	DefaultLeadTimeOrder	Number, Single	4	5	No	No
	DefaultLeadTimeShip	Number, Single	4	6	No	No
	DefaultShipCost	Number, Single	4	7	No	No
	ProgramContact	Text	50	8	No	No
	Telephone	Text	50	9	No	No
	Fax	Text	50	10	No	No
	Email	Text	50	11	No	No
	CountryName	Text	50	12	No	No
	IsCurrent	Yes/No	1	13	No	No
	Note	Memo		14	No	No
	ProgramCode	Text	50	15	No	Yes (Duplicates OK)

TableName	FieldName	DataType	FieldSize	Position	Primary Index	Secondary Index
	IsActive	Yes/No	1	16	No	No
	StartSize	Text	50	17	No	No
tmpPTReportTemp	ProductID	Text	10	1	No	Yes (Duplicates OK)
	ProductName	Text	50	2	No	No
	DesStock	Number, Double	8	3	No	No
tmpShipImp_Datasources	DataSourceID	Text	10	1	No	Yes (No Duplicates)
	DataSourceName	Text	50	2	No	No
	DataSourceTypeID	Text	10	3	No	Yes (Duplicates OK)
tmpShipImp_Fundingsources	FundingSourceID	Text	10	1	No	Yes (No Duplicates)
	FundingSourceName	Text	50	2	No	No
tmpShipImp_Products	ProductID	Text	10	1	No	Yes (No Duplicates)
	ProductName	Text	50	2	No	No
	ProductMinMonths	Number, Byte	1	3	No	No
	ProductMaxMonths	Number, Byte	1	4	No	No
	MethodID	Text	10	5	No	Yes (Duplicates OK)
	DefaultCaseSize	Number, Long Integer	4	6	No	No
	ProductNote	Memo		7	No	No
	ProdCMax	Number, Byte	1	8	No	No
	ProdCMin	Number, Byte	1	9	No	No
	ProdDesStock	Number, Byte	1	10	No	No
tmpShipImp_Shipments	ShipmentID	Number, Long Integer	4	1	No	Yes (Duplicates OK)
	ProductID	Text	10	2	No	Yes (Duplicates

TableName	FieldName	DataType	FieldSize	Position	Primary Index	Secondary Index
						OK)
	SupplierID	Text	10	3	No	Yes (Duplicates OK)
	ShipAmount	Number, Double	8	4	No	No
	ShipPlannedDate	Date/Time	8	5	No	No
	ShipReceivedDate	Date/Time	8	6	No	No
	ShipStatusCode	Text	1	7	No	Yes (Duplicates OK)
	ShipNote	Memo		8	No	No
	ShipDateChanged	Date/Time	8	9	No	No
	ShipFreightCost	Number, Double	8	10	No	No
	ShipValue	Number, Double	8	11	No	No
	ShipCaseLot	Number, Long Integer	4	12	No	No
	ShipDisplayNote	Yes/No	1	13	No	No
	ShipPO	Text	50	14	No	No
	strMapping	Text	50	15	No	No
	strMappingFull	Text	70	16	No	No
	fLocked	Yes/No	1	17	No	No
	fDataSourceID	Yes/No	1	18	No	No
	fSupplierID	Yes/No	1	19	No	No
	fProductID	Yes/No	1	20	No	No
	ProductName	Text	50	21	No	No
	fMapping	Yes/No	1	22	No	No
	dtmExported	Date/Time	8	23	No	No
	txtSource	Text	255	24	No	No
	ShipDataSourceID	Text	50	25	No	Yes (Duplicates OK)
	ShipFundingSourceID	Text	50	26	No	Yes (Duplicates OK)

TableName	FieldName	DataType	FieldSize	Position	Primary Index	Secondary Index
	fFundingSourceID	Yes/No	1	27	No	No
	lngID	Number, Long Integer	4	28	No	Yes (No Duplicates)
	fSplit	Yes/No	1	29	No	No
	fProcessed	Yes/No	1	30	No	No
tmpShipImp_Suppliers	SupplierID	Text	10	1	No	Yes (No Duplicates)
	SupplierName	Text	50	2	No	Yes (No Duplicates)
	SupplierLeadTimePlan	Number, Single	4	3	No	No
	SupplierLeadTimeOrder	Number, Single	4	4	No	No
	SupplierLeadTimeShip	Number, Single	4	5	No	No
	DefaultSupplier	Yes/No	1	6	No	Yes (Duplicates OK)
tmpShipments	ProductID	Text	10	1	No	Yes (Duplicates OK)
	SupplierID	Text	10	2	No	Yes (Duplicates OK)
	ShipDataSourceID	Text	10	3	No	Yes (Duplicates OK)
	ShipAmount	Number, Double	8	4	No	No
	ShipPlannedDate	Date/Time	8	5	No	No
	ShipOrderedDate	Date/Time	8	6	No	No
	ShipShippedDate	Date/Time	8	7	No	No
	ShipReceivedDate	Date/Time	8	8	No	No
	ShipStatusCode	Text	1	9	No	Yes (Duplicates OK)
	ShipNote	Memo		10	No	No
	ShipDateChanged	Date/Time	8	11	No	No

TableName	FieldName	DataType	FieldSize	Position	Primary Index	Secondary Index
	ShipFreightCost	Number, Single	4	12	No	Yes (Duplicates OK)
	ShipValue	Number, Double	8	13	No	Yes (Duplicates OK)
	ShipCaseLot	Number, Long Integer	4	14	No	No
	ShipDisplayNote	Yes/No	1	15	No	Yes (Duplicates OK)
	ShipPO	Text	50	16	No	Yes (Duplicates OK)
tmpSortID	CategoryID	Number, Long Integer	4	1	Yes	Yes (No Duplicates)
	SortID	Text	255	2	No	Yes (Duplicates OK)
tmpSuppliers	SupplierID	Text	10	1	No	Yes (No Duplicates)
	SupplierName	Text	50	2	No	Yes (No Duplicates)
	SupplierLeadTimePlan	Number, Single	4	3	No	No
	SupplierLeadTimeOrder	Number, Single	4	4	No	No
	SupplierLeadTimeShip	Number, Single	4	5	No	No
	SupplierNote	Memo		6	No	No
	Freight	Number, Single	4	7	No	No
	DefaultSupplier	Yes/No	1	8	No	Yes (Duplicates OK)
tmpTypeChanges	MethodID	Text	10	1	No	Yes (No Duplicates)
	MethodName	Text	50	2	No	Yes (No Duplicates)
	CYPFactor	Number, Single	4	3	No	No
	MethodNote	Memo		4	No	No

TableName	FieldName	DataType	FieldSize	Position	Primary Index	Secondary Index
	fRollup	Yes/No	1	5	No	No
	ParentID	Text	255	6	No	Yes (Duplicates OK)
	CategoryID	Number, Long Integer	4	7	No	Yes (No Duplicates)
tmpXtabStockStatusMatrix	KeyID	Text	10	1	No	No
	KeyName	Text	50	2	No	No
	Unit	Text	25	3	No	No
	StockMin	Number, Byte	1	4	No	No
	StockMax	Number, Byte	1	5	No	No
	MOS	Number, Double	8	6	No	No
	StockDate	Date/Time	8	7	No	No
	StockShipSupplier	Text	10	8	No	No
	AMCFlag	Number, Long Integer	4	9	No	No
TranslateText	TextID	Text	6	1	Yes	Yes (Duplicates OK)
	EnglishText	Text	255	2	No	No
	FrenchText	Text	255	3	No	No
	SpanishText	Text	255	4	No	No
	ArabicText	Text	255	5	No	No
	Comment	Text	50	6	No	No
	NeedsTranslation	Yes/No	1	7	No	Yes (Duplicates OK)
	PortugueseText	Text	255	8	No	No
XtabResult	Column0	Text	50	1	No	No
	Column1	Text	50	2	No	No
	Column2	Currency	8	3	No	No
	Column3	Currency	8	4	No	No

TableName	FieldName	DataType	FieldSize	Position	Primary Index	Secondary Index
	Column4	Currency	8	5	No	No
	Column5	Currency	8	6	No	No
	Column6	Currency	8	7	No	No
	Column7	Currency	8	8	No	No
	Column8	Currency	8	9	No	No
	Column9	Currency	8	10	No	No
XtabResultMulti	Column0	Text	50	1	No	No
	Column1	Text	50	2	No	No
	Column2	Text	50	3	No	No
	Column3	Currency	8	4	No	No
	Column4	Currency	8	5	No	No
	Column5	Currency	8	6	No	No
	Column6	Currency	8	7	No	No
	Column7	Currency	8	8	No	No
	Column8	Currency	8	9	No	No
	Column9	Currency	8	10	No	No
	Column10	Currency	8	11	No	No
XtabResultStockStatus	Column0	Text	50	1	No	No
	Column1	Text	50	2	No	No
	Column2	Number, Byte	1	3	No	No
	Column3	Number, Byte	1	4	No	No
	Column4	Number, Single	4	5	No	No
	Column5	Number, Single	4	6	No	No
	Column6	Number, Single	4	7	No	No
	Column7	Number, Single	4	8	No	No
	Column8	Number, Single	4	9	No	No
	Column9	Number, Single	4	10	No	No

TableName	FieldName	DataType	FieldSize	Position	Primary Index	Secondary Index
	Column10	Number, Single	4	11	No	No
	Column11	Number, Single	4	12	No	No
	Column12	Number, Single	4	13	No	No
	Column13	Number, Single	4	14	No	No
	Column14	Number, Single	4	15	No	No
	Column15	Number, Single	4	16	No	No
	Column16	Number, Single	4	17	No	No

Program Units

Overview

PipeLine was developed using Microsoft Access based on a split database design. Under this structure the application interface (forms, queries, reports, lookup tables, and code) reside in one database called the front-end and the data database, called the backend.

Source Code CD content

The source code cd contains the application required files, plus installer files, and documentation to assist the developer in understanding PipeLine.

Table 16 - Source Code CD Content

Directory/File	**Purpose**
PL- Module Printout.pdf	Module Printout
PL – Procedure List.pdf	Procedure List
/PipeLine	**Parent directory**
/CODE	**Code directory**
/ANYMOH	**Directory for sample database**
/globalmoh.MDB	Sample Database
/Graphics	**Directory for application graphics**
/SplashNewT.avi	Splash screen movie
/PL40.ico	PipeLine taskbar icon
/Pipeline_ICON-xx.ico	PipeLine Desktop icon
/Summary	**Directory for PipeLine Summary**
/Roboex32.dll	Dll required for PipeLine Summary to run properly
/Proc2000.mdb	PipeLine Summary frontend
/Proc_BE.mdb	PipeLine Summary backend
/Prog2000.mdb	PipeLine Summary program list
/Summary.ico	PipeLine Summary icon
/Sumv2.cnt	PipeLine Summary help cnt file
/SUMv2.hlp	PipeLine Summary help file
/XML	**Directory for xml files**
/ECatalog_Live_Final_Generic_20100701.xml	E-Catalog file distributed with application
/SCMS Product_ARV_TEST.xml	SCMS ARV file distributed with application
/Contraceptives.xml	Contraceptives file distributed with application
/e-help.cnt	PipeLine (English) help cnt file
/e-help.HLP	PipeLine (English) help file
/E-PL-help.cnt	PipeLine help cnt file

Directory/File	Purpose
/E-PL-help.hlp	PipeLine help file
/Pipeline_ICON-xx.ico	PipeLine icon
/Pipeline2000.MDB	PipeLine frontend file
/PLFix1.reg	Registry fix for graphs
/PLFix2.reg	Registry fix for graphs
/PLFix3.reg	Registry fix for graphs
/pmp_mpty.mdb	Empty PipeLine backend file
/posttransform.xslt	
/ProgV4.mdb	PipeLine program list
/ReadMe.txt	Readme file for installation issues and known issues
/Roboex32.dll	Dll required for PipeLine to run properly
/Help Files	**Help file directory**
/PipeLine	**Pipeline directory**
/English	**English directory**
/E-PL-Help.doc	Help word document
/E-PL-Help.hpj	Help project
/!SSL!	**Build directory**
/WinHelp2000	**WinHelp2000 directory**
/e-help.cnt	context file
/e-help.hlp	Help file
/e-pl-help.cnt	context file
/ e-pl-help.hlp	Help file
/ e-pl-help.log	Log file
/French	**French directory**
/E-PL-Help.doc	Help word document
/E-PL-Help.hpj	Help project
/Spanish	**Spanish directory**
/E-PL-Help.doc	Help word document
/E-PL-Help.hpj	Help project
/Sumv2	**PL Summary directory**
/Sumv2.doc	Help word document
/ Sumv2.hpj	Help project
/Installer	**Installer directory**
/Distribution	**Distribution directory**
/accessrt.cab	Access runtime files
/Accessrt.msi	Access runtime installer
/addendum to user manual for pipeline 5 – English.pdf	Addendum 5.0
/autorun.aru	Autorun file
/autorun.ico	Autorun icon
/autorun.inf	Autorun ini ile
/data1.cab	Added references for installer

Directory/File	Purpose
/install.ini	Installer ini file
/msxml6.msi	XML 6 installer
/new features and known issues in PipeLine 5.1.pdf	Addendum for 5.1
/pipeline5.cab	Pipeline installer files
/pipeline user manual 4 – English.pdf	User Guide for 4.0
/pipeline.msi	Pipeline installer
/Pipeline5_1.exe	Exe that run msi file
/Readme.txt	Readme text for installer
/setup.exe	Setup file for complete installer
/Web	**Web directory**
/About.txt	Text for about statement
/PipeLine5_1Master.zip	Master zip file of installer
/password.txt	Text for password warning
/pipeline5_1.exe	Exe file created by winzip
/pipeline5_1.zip	Zip file containing exe and readme file
/readme.txt	Readme file for installation
/Schemas	**Schema directory**
/MaterialMasterNorm_070516_NoCustomType.xsd	Material Master schema
/SCMgr_PipeLine_Export.xsd	SCMgr schema
/QuantimedForecastOutput.xsd	Quantimed schema
/PipelineXMLOutputSchema_070924.xsd	XML export schema

Program name

The complete Module/Procedure List can be found at:

PL - Procedure List.pdf

The Complete Source code:

PL - Module Printout.pdf

Forms

Table 17 - List of Forms

ModuleName	Procedure	Lines	Comments
fdlgCloneConsumptionSelectDate	(General Declarations)	1 - 18	' Module : fdlgSelectDateRange ' Description: ' Procedures : cmdCancel_Click ' cmdOK_Click ' Form_Open
	BuildFromYears	224 - 268	
	cmdCancel_Click	87 - 107	' Comments : Close the form, which returns the focus ' : to the calling form. ' Parameters : ' Created : 17-Feb-99 Jeff Leiner ' Modified : ' ' --
	cmdOK_Click	108 - 175	' Comments : Check that the data is OK and then set the ' : form's visibility to false to return the focus ' : to the calling form. ' Parameters : ' Created : 17-Feb-99 Jeff Leiner ' Modified : ' ' --
	EndYear	70 - 81	
	Form_Open	176 - 211	' Comments : Look up the current month and set ' : Date range to a 6 month default. ' Parameters : ' Created : ' Modified : ' ' --
	OK	27 - 37	'+ ' Property(G): OK ' Comments : ' Parameters : ' Returns : Boolean ' Created : 23 Jul 2009 jleiner ' Modified : '-
	StartDate	49 - 60	
fdlgCloneConsumptionSetRatio	(General Declarations)	1 - 7	
	CloseMe	526 - 535	
	cmdCancel_Click	29 - 35	
	cmdClone_Click	37 - 55	
	cmdPreview_Click	58 - 69	
	dblFactor_AfterUpdate	73 - 81	
	Form_GotFocus	83 - 87	
	Form_Load	95 - 116	
	Form_Open	520 - 524	
	OK	16 - 26	'+ ' Property(G): OK ' Comments : ' Parameters : ' Returns : Boolean '

ModuleName	Procedure	Lines	Comments
			Created : 17 Jul 2009 jleiner ' Modified : '-
	ProduceConsumptionRecords	128 - 253	
	SaveConsumptionRecords	266 - 518	
fdlgCountryCode	(General Declarations)	1 - 2	
	cmdCancel_Click	9 - 22	'+ ' Procedure : cmdCancel_Click ' Comments : closes form and sends user back to previous screen ' Parameters: - ' Modified : 16 Mar 2007 MAT '-
	cmdOK_Click	45 - 74	'place country code into program data table
	Form_Open	30 - 43	
fdlgCPTGenerator	(General Declarations)	1 - 21	' Module : Form_fdlgCPTGenerator ' Description: ' Procedures : cboYear_AfterUpdate() ' cmdCancel_Click() ' cmdCreateCPTData_Click() ' Form_Activate() ' Form_Load() ' Form_Open(Cancel As Integer) ' SetComboYears() ' SetText()
	cboYear_AfterUpdate	22 - 46	' Comments : ' Parameters : ' Returns : ' Created : ' Modified : 01 Jul 1998 JSL ' ' ---
	cmdCancel_Click	48 - 67	' Comments : ' Parameters : ' Returns : ' Created : ' Modified : 01 Jul 1998 JSL ' ' ---
	cmdCreateCPTData_Click	69 - 356	' Comments : To Generate the CPT Data files for export to CPT and NewVern. ' : some files will be generated just from the tables data others will ' : be generated due to the products selected in the lstProducts listbox ' Parameters : ' Returns : ' Created : 05/26/98 JSL * ' Modified : 01 Jul 1998 JSL '**
	Form_Activate	358 - 377	' Comments : ' Parameters : ' Returns : ' Created : ' Modified : 01 Jul 1998 JSL ' ' ---
	Form_Load	379 - 398	' Comments : ' Parameters : ' Returns : ' Created : ' Modified : 01 Jul 1998 JSL ' ' ---
	Form_Open	400 - 419	' Comments : ' Parameters : Cancel ' Returns : ' Created : ' Modified : 01 Jul 1998 JSL ' ' ---
	SetComboYears	421 - 474	' Comments : To set the range of years to place in the combobox for the CPT ' report. ' Parameters : ' Returns : ' Created : 05/26/98 JSL ' Modified : 01 Jul 1998 JSL ' ' ---
	SetText	476 - 496	' Comments : ' Parameters : ' Returns : ' Created : ' Modified : 01 Jul 1998

ModuleName	Procedure	Lines	Comments
			JSL ' ' --
fdlgExportCYP	(General Declarations)	1 - 14	' Module : Form_fdlgExportCYP ' Description: ' Procedures : SetText() ' cmdCancel_Click() ' cmdOK_Click() ' Form_Open(Cancel As Integer)
	cmdCancel_Click	34 - 51	' Comments : ' Parameters: - ' Modified : ' ' --
	cmdOK_Click	53 - 62	' Comments : ' Parameters: - ' Modified : ' ' --
	Form_Open	64 - 73	' Comments : ' Parameters: Cancel - ' Modified : ' ' --
	SetText	15 - 32	' Comments : ' Parameters: - ' Modified : ' ' --
fdlgForecastMappingSCMS	(General Declarations)	1 - 18	'+ ' Module : Form_fdlgForecastMappingSCMS ' Description: Used for mapping products from SCMS source to PipeLine ' Procedures : ClearData() ' ClearProcessed() ' cmdCancel_Click() ' cmdOK_Click() ' Form_Open(Cancel As Integer)
	cboSCMSMapping_AfterUpdate	44 - 71	'this may replace _beforeUpdate just below
	cboSCMSMapping_BeforeUpdate	73 - 91	'if this has a value when the form loads then 'it has to still have a value after update 'in other words, it can't be changed to null if it had value to begin with 'mtracy oct 2006 'Dim fSCMSMapping As Boolean 'fSCMSMapping = False 'default value
	cboSCMSMapping_GotFocus	93 - 100	
	cboSCMSMapping_LostFocus	104 - 106	
	ClearUnMapped	25 - 42	'+ ' Procedure : ClearUnMapped ' Comments : Removes all unmapped data from the Import tables ' Parameters: - ' Modified : 17 Jul 2006 MAT '-
	cmdCancel_Click	114 - 133	
	cmdOK_Click	141 - 219	
	Form_Open	226 - 239	'+ ' Procedure : Form_Open ' Comments : move to first item in source lookup up and set text ' Parameters: Cancel - ' Modified : 14 Jul 2006 MAT '-
	ProductNameCut	289 - 299	
	txtMappingFull_Click	244 - 246	
	txtPLMappingFull_Click	241 - 243	

ModuleName	Procedure	Lines	Comments
	txtPLProduct_AfterUpdate	247 - 274	
	txtPLProduct_GotFocus	276 - 280	'requery all the dropdowns, but don't show items in text boxes
	txtPLProduct_LostFocus	301 - 303	
fdlgImportForecast	(General Declarations)	1 - 24	'+ ' Module : Form_fdlgImportForecast ' Description: ' Procedures : cmdBrowse_Click() ' cmdCancel_Click() ' cmdOK_Click() ' Form_Close() ' Form_Open(Cancel As Integer)
	cmdBrowse_Click	31 - 52	'+ ' Procedure : cmdBrowse_Click ' Comments : Opens form for user to select a program file to open ' Parameters: - ' Modified : 27 Jun 2003 LBlanken '-
	cmdBrowse2_Click	60 - 81	
	cmdCancel_Click	89 - 105	
	cmdOK_Click	113 - 188	
	Form_Open	196 - 214	
fdlgImportMapping	(General Declarations)	1 - 19	'+ ' Module : Form_fdlgImportMapping ' Description: Used for mapping products for external source to PipeLine ' Procedures : cboSelect1_AfterUpdate() ' ClearData() ' ClearProcessed() ' cmdCancel_Click() ' cmdOK_Click() ' Form_Open(Cancel As Integer) ' ResetValues()
	cboSelect1_AfterUpdate	26 - 38	'+ ' Procedure : cboSelect1_AfterUpdate ' Comments : Filters based on source selected ' Parameters: - ' Modified : 21 Aug 2003 LKB '-
	ClearData	46 - 69	
	ClearProcessed	77 - 94	
	cmdCancel_Click	102 - 124	
	cmdOK_Click	132 - 281	
	Form_Open	289 - 313	
	ResetValues	321 - 336	
	Validate_Period	350 - 442	
fdlgImportMappingSCMS	(General Declarations)	1 - 16	'+ ' Module : Form_fdlgImportMappingSCMS ' Description: Used for mapping products from SCMS source to PipeLine ' Procedures : ClearData() ' ClearProcessed() ' cmdCancel_Click() ' cmdOK_Click() ' Form_Open(Cancel As Integer)

ModuleName	Procedure	Lines	Comments
	cmdCancel_Click	23 - 42	'+ ' Procedure : cmdCancel_Click ' Comments : resets all values to null (mapping/clear flag) and closes form ' Parameters: - ' Modified : 21 Aug 2003 LKB '-
	cmdOK_Click	50 - 55	
	Form_Open	62 - 75	'+ ' Procedure : Form_Open ' Comments : move to first item in source lookup up and set text ' Parameters: Cancel - ' Modified : 14 Jul 2006 MAT '-
	txtMappingFull_Click	77 - 79	
	txtProduct_AfterUpdate	81 - 104	
	txtProduct_GotFocus	106 - 110	'requery all the dropdowns, but don't show items in text boxes
fdlgImportProducts	(General Declarations)	1 - 24	'+ ' Module : Form_fdlgImportProducts ' Description: ' Procedures : cmdBrowse_Click() ' cmdCancel_Click() ' cmdOK_Click() ' Form_Close() ' Form_Open(Cancel As Integer)
	cmdBrowse_Click	31 - 52	'+ ' Procedure : cmdBrowse_Click ' Comments : Opens form for user to select a program file to open ' Parameters: - ' Modified : 27 Jun 2003 LBlanken '-
	cmdCancel_Click	60 - 76	
	cmdOK_Click	84 - 139	
	Form_Open	147 - 164	
fdlgImportSCM	(General Declarations)	1 - 24	'+ ' Module : Form_fdlgImportSCM ' Description: ' Procedures : cmdBrowse_Click() ' cmdCancel_Click() ' cmdOK_Click() ' Form_Close() ' Form_Open(Cancel As Integer)
	cmdBrowse_Click	31 - 52	'+ ' Procedure : cmdBrowse_Click ' Comments : Opens form for user to select a program file to open ' Parameters: - ' Modified : 27 Jun 2003 LBlanken '-
	cmdCancel_Click	60 - 76	
	cmdOK_Click	84 - 111	
	Form_Close	119 - 138	
	Form_Open	146 - 159	
fdlgImportShipmentsPL	(General Declarations)	1 - 23	'+ ' Module : Form_fdlgShipmentsPL ' Description: ' Procedures : cmdBrowse_Click() ' cmdCancel_Click() ' cmdOK_Click() ' Form_Close() ' Form_Open(Cancel As Integer) ' Created : 1 Sep 2009 mahmed '

ModuleName	Procedure	Lines	Comments
			Modified : '-
	cmdBrowse_Click	30 - 51	'+ ' Procedure : cmdBrowse_Click ' Comments : Opens form for user to select a program file to open ' Parameters: - ' Modified : '-
	cmdCancel_Click	59 - 75	
	cmdOK_Click	83 - 117	
	Form_Open	125 - 139	
fdlgInterpolate	(General Declarations)	1 - 33	' Module : Form_fdlgCPTGenerator ' Description: ' Procedures : cboYear_AfterUpdate() ' cmdCancel_Click() ' cmdCreateCPTData_Click() ' Form_Activate() ' Form_Load() ' Form_Open(Cancel As Integer) ' SetComboYears() ' SetText()
	cboMonthStarting_AfterUpdate	34 - 42	
	cboYearEnding_AfterUpdate	44 - 50	
	cboYearStarting_AfterUpdate	52 - 60	
	cmdApply_Click	63 - 95	
	cmdCancel_Click	101 - 136	' Comments : ' Parameters : ' Returns : ' Created : ' Modified : ' ' --
	cmdPreview_Click	138 - 172	
	DisplayReport	618 - 634	
	Form_Activate	174 - 193	' Comments : ' Parameters : ' Returns : ' Created : ' Modified : ' ' --
	Form_Load	195 - 214	' Comments : ' Parameters : ' Returns : ' Created : ' Modified : ' ' --
	Form_Open	216 - 240	' Comments : ' Parameters : Cancel ' Returns : ' Created : ' Modified : ' ' --
	lngAMCEnding_AfterUpdate	344 - 350	
	lngAMCStarting_AfterUpdate	360 - 366	
	SaveData	380 - 594	
	SetComboYears	248 - 312	' Comments : To set the range of years to place in the combobox for the CPT ' report. ' Parameters : ' Returns : ' Created : ' Modified : ' ' --
	SetText	314 - 334	' Comments : ' Parameters : ' Returns : ' Created : ' Modified : ' ' ------------

ModuleName	Procedure	Lines	Comments
			--
fdlgLanguage	(General Declarations)	1 - 17	' Module : Form_fdlgLanguage ' Description: ' Procedures : cboSelect_AfterUpdate() ' cmdAdd_Click() ' cmdCancel_Click() ' Form_Open(Cancel As Integer) ' SetText()
	cboSelect_AfterUpdate	18 - 37	' Comments : ' Parameters : ' Returns : ' Created : ' Modified : 01 Jul 1998 JSL ' ' --
	cmdAdd_Click	39 - 102	' Comments : ' Parameters : ' Returns : ' Created : ' Modified : 01 Jul 1998 JSL ' ' --
	cmdCancel_Click	104 - 130	' Comments : ' Parameters : ' Returns : ' Created : ' Modified : 01 Jul 1998 JSL ' ' --
	Form_Open	131 - 152	' Comments : ' Parameters : Cancel ' Returns : ' Created : ' Modified : 01 Jul 1998 JSL ' ' --
	SetText	154 - 181	' Comments : ' Parameters : ' Returns : ' Created : ' Modified : 01 Jul 1998 JSL ' ' --
fdlgOpenProgram	(General Declarations)	1 - 23	'+ ' Module : Form_fdlgOpenProgram ' Description: Form allows user to select an inactive form to make active ' Procedures : cmdBrowse_Click() ' cmdCancel_Click() ' cmdOK_Click() ' Form_Open(Cancel As Integer) ' lstSelect_AfterUpdate()
	cmdBrowse_Click	30 - 53	'+ ' Procedure : cmdBrowse_Click ' Comments : Opens form for user to select a program file to open ' Parameters: - ' Modified : 27 Jun 2003 LBlanken '-
	cmdCancel_Click	61 - 76	
	cmdOK_Click	84 - 106	
	Form_Open	114 - 129	
	lstSelect_AfterUpdate	137 - 151	
fdlgPlanShipmentsByDate	(General Declarations)	1 - 17	'+ ' Module : Form_fdlgPlanShipmentsByDate ' Description: Creates shipments records for products and date entered on form ' Procedures : cmdCancel_Click() ' cmdOK_Click() ' GetCaseLot(strProductID As String, strSupplierID As String)
	cmdCancel_Click	24 - 37	'+ ' Procedure : cmdCancel_Click ' Comments : Close form w/o doing anything ' Parameters: - ' Modified : 27 Jun 2003 LBlanken '-
	cmdOK_Click	46 - 161	
	Form_Error	267 - 300	

ModuleName	Procedure	Lines	Comments
	Form_Open	302 - 304	
	GetCaseLot	169 - 238	
	UpdateDate	239 - 265	
fdlgProductErrors	(General Declarations)	1 - 13	'+ ' Module : Form_fdlgProductErrors ' Description: Used for making sure imported product names are 50 characters or less ' Procedures : cmdCancel_Click() ' cmdOK_Click() ' Form_Open(Cancel As Integer)
	cmdCancel_Click	20 - 41	'+ ' Procedure : cmdCancel_Click ' Comments : resets values, closes form, and sends user back to previous screen ' Parameters: - ' Modified : 9 Feb 2007 MAT '-
	cmdOK_Click	49 - 65	
	Form_Open	72 - 90	'+ ' Procedure : Form_Open ' Comments : move to first item in source lookup up and set text ' Parameters: Cancel - ' Modified : 14 Jul 2006 MAT '-
	txtProductNew_AfterUpdate	92 - 94	'Me.strName = Me.txtProductNew
	txtProductNew_BeforeUpdate	96 - 98	'Me.txtProductNew = Left([strName], 50)
fdlgProperties	(General Declarations)	1 - 15	' Module : Form_frmAbout ' Description: ' Procedures : cboLanguage_AfterUpdate() ' Form_Open(Cancel As Integer) ' Form_Timer() ' SetText()
	cmdClose_Click	16 - 18	
	Form_Open	20 - 43	' Comments : On Open see if an upgrade is needed. If yes ' : give the user the option, if not act as a splash ' : screen. ' Parameters : ' Returns : ' Created : 06/29/98 JSL ' Modified : ' ' --
	Form_Timer	45 - 69	' Comments : if TimerInterval > 0 then count the iterations and ' : do do the correct events ' Parameters : ' Returns : ' Created : 06/29/98 JSL ' Modified : ' ' --
	SetText	71 - 94	' Comments : Display the text in the proper language ' Parameters : ' Returns : ' Created : ' Modified : 01 Jul 1998 JSL ' ' --
fdlgRecalcAMC	(General Declarations)	1 - 24	'+ ' Module : Form_fdlgStockCount ' Description: Form displays when user click on Enter Count button ' from the stock form. Allows the user to input an ' inventory count and will calculate the adjustment ' based on that value. ' Procedures : cmdCancel_Click() ' cmdRecalc_Click() '

ModuleName	Procedure	Lines	Comments
			cmdSave_Click() ' Form_Open(Cancel As Integer)
	cmdCancel_Click	32 - 49	'+ ' Procedure : cmdCancel_Click ' Comments : Run when user clicks on cancel button ' Parameters: - ' Modified : 03 Jun 2003 LBlanken ' 16 Jun 2003 LBlanken '-
	cmdSave_Click	59 - 111	
	Form_Error	113 - 160	'+ ' Procedure : Form_Error ' Comments : Handles validation Errors ' Parameters: DataErr ' Response - ' Modified : 01 Jul 1998 JSL ' 03 Jun 2003 LBlanken ' 16 Jun 2003 LBlanken '-
	Form_Open	169 - 221	
fdlgSelectDateRange	(General Declarations)	1 - 12	' Module : fdlgSelectDateRange ' Description: ' Procedures : cmdCancel_Click ' cmdOK_Click ' Form_Open
	BuildFromYears	132 - 160	
	BuildThroughYears	171 - 199	'+ ' Procedure : BuildThroughYears ' Comments : ' Parameters: strType ' frmin ' strIN ' intAdd - ' Returns : - ' Modified : 12 May 2004 LKB ' -
	cmdCancel_Click	13 - 33	' Comments : Close the form, which returns the focus ' : to the calling form. ' Parameters : ' Created : 17-Feb-99 Jeff Leiner ' Modified : ' ' --
	cmdOK_Click	34 - 88	' Comments : Check that the data is OK and then set the ' : form's visibility to false to return the focus ' : to the calling form. ' Parameters : ' Created : 17-Feb-99 Jeff Leiner ' Modified : ' ' --
	Form_Open	89 - 119	' Comments : Look up the current month and set ' : Date range to a 6 month default. ' Parameters : ' Created : ' Modified : ' ' --
	txtStartYear_AfterUpdate	202 - 225	
fdlgSelectPDFPrinter	(General Declarations)	1 - 2	
	cmdCancel_Click	3 - 8	
	cmdSetPrinter_Click	10 - 20	
	Form_Open	22 - 27	
	PopulatePrinterList	29 - 53	
fdlgSelectRecipient	(General Declarations)	1 - 15	' Module : Form_fdlgSelectRecipient ' Description: ' Procedures : SetText() ' cmdCancel_Click() ' cmdOK_Click() ' Form_Open(Cancel As

ModuleName	Procedure	Lines	Comments
			Integer) ' lstRecipients_Click()
	cmdCancel_Click	32 - 41	' Comments : ' Parameters: - ' Modified : ' ' --
	cmdOK_Click	43 - 66	' Comments : ' Parameters: - ' Modified : ' ' --
	Form_Open	68 - 77	' Comments : ' Parameters: Cancel - ' Modified : ' ' --
	lstRecipients_Click	79 - 88	' Comments : ' Parameters: - ' Modified : ' ' --
	SetText	16 - 30	' Comments : ' Parameters: - ' Modified : ' ' --
fdlgShipmentMapping	(General Declarations)	1 - 35	
	checkMappingUsed	77 - 107	
	chkAllShipments_AfterUpdate	36 - 66	
	chkSelect_AfterUpdate	122 - 128	
	chkSplit_AfterUpdate	138 - 146	
	cmdCancel_Click	155 - 172	
	cmdDetail_Click	174 - 189	
	cmdOK_Click	199 - 207	'Save record
	cmdReport_Click	210 - 216	
	cmdShipmentDetail_Click	332 - 344	
	Command145_Click	348 - 362	
	Form_Close	218 - 220	'AppCloseEnabled (True)
	Form_Open	229 - 250	
	txtMappingFull_Click	252 - 259	
	txtProduct_AfterUpdate	262 - 310	'End Sub '+ ' Procedure : txtProduct_Change() ' Comments : reloads all the other dropdowns on the form ' also fills in text box value for said record ' Parameters: - ' Modified : 1 Sep 2009 mahmed '-
	txtProduct_GotFocus	313 - 322	
fdlgShipmentSearch	(General Declarations)	1 - 24	'+ ' Module : Form_fdlgShipmentSearch ' Description: ' Procedures : cmdSearch_Click() ' cmdCancel_Click() ' cmdNext_Click() ' Form_Close()

ModuleName	Procedure	Lines	Comments
			' Form_Open(Cancel As Integer)
	cmdCancel_Click	31 - 48	'+ ' Procedure : cmdCancel_Click ' Comments : Closes the form and returns to shipment module ' Parameters: - ' Modified : 28 July 2009 MTracy '-
	cmdNext_Click	50 - 78	'this sub just moves us ahead one rec in the search results 'and then reloads the shipment form with the new product/shipment highlighted
	cmdSearch_Click	86 - 198	
	Form_Open	206 - 222	
fdlgStockCount	(General Declarations)	1 - 23	'+ ' Module : Form_fdlgStockCount ' Description: Form displays when user click on Enter Count button ' from the stock form. Allows the user to input an ' inventory count and will calculate the adjustment ' based on that value. ' Procedures : cmdCancel_Click() ' cmdRecalc_Click() ' cmdSave_Click() ' Form_Open(Cancel As Integer)
	cmdCancel_Click	31 - 45	'+ ' Procedure : cmdCancel_Click ' Comments : Run when user clicks on cancel button ' Parameters: - ' Modified : 03 Jun 2003 LBlanken ' 16 Jun 2003 LBlanken '-
	cmdRecalc_Click	54 - 69	
	cmdSave_Click	78 - 122	
	Form_Error	124 - 173	'+ ' Procedure : Form_Error ' Comments : Handles validation Errors ' Parameters: DataErr ' Response - ' Modified : 01 Jul 1998 JSL ' 03 Jun 2003 LBlanken ' 16 Jun 2003 LBlanken '-
	Form_Open	182 - 213	
fdlgTextAdmin	(General Declarations)	1 - 2	
	ArabicStatus_Change	22 - 24	
	cmdClose_Click	3 - 5	
	EnglishStatus_Change	7 - 9	
	Form_Open	11 - 13	
	FrenchStatus_Change	15 - 17	
	portugueseStatus_Change	25 - 27	
	SpanishStatus_Change	19 - 21	
	txtDocumentType_AfterUpdate	28 - 32	

ModuleName	Procedure	Lines	Comments
	txtFormName_AfterUpdate	34 - 38	
fdlgUpdateProducts	(General Declarations)	1 - 26	'+ ' Module : Form_fdlgUpdateProducts ' Description: ' Procedures : cmdBrowse_Click() ' cmdCancel_Click() ' cmdOK_Click() ' Form_Close() ' Form_Open(Cancel As Integer)
	cmdBrowse_Click	33 - 54	'+ ' Procedure : cmdBrowse_Click ' Comments : Opens form for user to select a program file to open ' Parameters: - ' Modified : 27 Jun 2003 LBlanken '-
	cmdCancel_Click	62 - 78	
	cmdOK_Click	86 - 150	
	Form_Open	158 - 175	
frmAbout	(General Declarations)	1 - 17	' Module : Form_frmAbout ' Description: ' Procedures : cboLanguage_AfterUpdate() ' Form_Open(Cancel As Integer) ' Form_Timer() ' SetText()
	cboLanguage_AfterUpdate	18 - 39	' Comments : When the language is modified, update the system ' : variables, and the display. ' Parameters : ' Returns : ' Created : ' Modified : 01 Jul 1998 JSL ' ' --
	cmdClose_Click	42 - 44	
	Form_Open	46 - 76	' Comments : On Open see if an upgrade is needed. If yes ' : give the user the option, if not act as a splash ' : screen. ' Parameters : ' Returns : ' Created : 06/29/98 JSL ' Modified : ' ' --
	Form_Timer	78 - 106	' Comments : if TimerInterval > 0 then count the iterations and ' : do do the correct events ' Parameters : ' Returns : ' Created : 06/29/98 JSL ' Modified : ' ' --
	SetText	108 - 141	' Comments : Display the text in the proper language ' Parameters : ' Returns : ' Created : ' Modified : 01 Jul 1998 JSL ' ' --
frmBlank	(General Declarations)	1 - 2	
	Form_Open	3 - 10	
frmCaseSizes	(General Declarations)	1 - 27	' Module : Form_frmCaseSizes ' Description: ' Procedures : cboSelect1_AfterUpdate() ' cboSelect2_AfterUpdate() ' cmdDelete_Click() ' cmdNew_Click() ' cmdSave_Click() ' Form_Activate() ' Form_BeforeUpdate(Cancel As Integer) ' Form_Delete(Cancel As

ModuleName	Procedure	Lines	Comments
			Integer) ' Form_Error(DataErr As Integer, Response As Integer) ' Form_Open(Cancel As Integer) ' lstSelect_Click() ' SetText()
	cboSelect1_AfterUpdate	28 - 58	' Comments : ' Parameters : ' Returns : ' Created : ' Modified : 01 Jul 1998 JSL ' ' --
	cboSelect1_GotFocus	60 - 82	' Comments : ' Parameters : ' Returns : ' Created : 08-Mar-00 Jeff Leiner ' Modified : ' ' --
	cboSelect2_AfterUpdate	84 - 141	' Comments : ' Parameters : ' Returns : ' Created : ' Modified : 01 Jul 1998 JSL ' ' --
	cboSelect2_GotFocus	143 - 165	' Comments : ' Parameters : ' Returns : ' Created : 08-Mar-00 Jeff Leiner ' Modified : ' ' --
	cmdDelete_Click	167 - 215	' Comments : ' Parameters : ' Returns : ' Created : ' Modified : 01 Jul 1998 JSL ' ' --
	cmdNew_Click	217 - 238	' Comments : ' Parameters : ' Returns : ' Created : ' Modified : 01 Jul 1998 JSL ' ' --
	cmdSave_Click	240 - 284	' Comments : ' Parameters: - ' Modified : ' ' --
	Form_Activate	286 - 313	' Comments : ' Parameters : ' Returns : ' Created : ' Modified : 01 Jul 1998 JSL ' ' --
	Form_BeforeUpdate	315 - 335	' Comments : ' Parameters : Cancel ' Returns : ' Created : ' Modified : 01 Jul 1998 JSL ' ' --
	Form_Delete	337 - 362	' Comments : ' Parameters : Cancel ' Returns : ' Created : ' Modified : 01 Jul 1998 JSL ' ' --
	Form_Dirty	364 - 366	
	Form_Error	368 - 411	' Comments : ' Parameters : DataErr ' Response ' Returns : ' Created : ' Modified : 01 Jul 1998 JSL ' ' --
	Form_Open	413 - 489	' Comments : ' Parameters : Cancel ' Returns : ' Created : ' Modified : 01 Jul 1998 JSL ' ' --
	lstSelect_Click	491 - 519	' Comments : ' Parameters : ' Returns : ' Created : 08-Mar-00 Jeff Leiner ' Modified : ' ' --
	MovetoClosestCaseSize	550 - 616	
	SetText	521 - 541	' Comments : To Set each label to the correct text. ' Parameters : ' Returns : ' Created : ' Modified : 01 Jul 1998 JSL ' : 05/18/98 JSL - modified Commodity form to create Case Size ' ------------------------------------

ModuleName	Procedure	Lines	Comments
			----------------------
frmCategory	(General Declarations)	1 - 26	'+ ' Module : Form_frmCategory ' Description: Allows users to manage categories throughout system ' Procedures : chkType_AfterUpdate() ' cmdDelete_Click() ' cmdNew_Click() ' cmdSave_Click() ' Form_Activate() ' Form_BeforeUpdate(Cancel As Integer) ' Form_Dirty(Cancel As Integer) ' Form_Error(DataErr As Integer, Response As Integer) ' Form_Open(Cancel As Integer) ' lstSelect_Click() ' SetText() ' txtName_AfterUpdate()
	chkType_AfterUpdate	33 - 46	'+ ' Procedure : chkType_AfterUpdate ' Comments : Enables the txtCoverage field if true ' Parameters: - ' Modified : 23 Jun 2003 LBlanken '-
	cmdDelete_Click	55 - 112	
	cmdNew_Click	121 - 157	
	cmdSave_Click	165 - 218	
	Form_Activate	226 - 242	
	Form_AfterUpdate	243 - 245	
	Form_BeforeInsert	246 - 248	
	Form_BeforeUpdate	256 - 291	
	Form_Close	294 - 303	
	Form_Dirty	311 - 324	
	Form_Error	334 - 368	
	Form_Open	377 - 452	
	lstSelect_Click	460 - 496	
	SetText	505 - 519	
	txtName_AfterUpdate	527 - 544	
	txtParent_AfterUpdate	546 - 573	
frmCosts	(General Declarations)	1 - 29	' Module : Form_frmCosts ' Description: ' Procedures : MarkCurrentPrice() ' cboSelect1_AfterUpdate() ' cboSelect2_AfterUpdate() ' cmdDelete_Click() ' cmdEdit_Click() ' cmdNew_Click() ' cmdSave_Click() ' Form_Activate() ' Form_BeforeUpdate(Cancel As Integer) ' Form_Delete(Cancel As Integer) ' Form_Error(DataErr As Integer, Response As Integer) '

ModuleName	Procedure	Lines	Comments
			Form_Open(Cancel As Integer) ' lstSelect_Click() ' SetText()
	cboSelect1_AfterUpdate	75 - 104	' Comments : ' Parameters : ' Returns : ' Created : ' Modified : 01 Jul 1998 JSL ' ' --
	cboSelect1_GotFocus	106 - 130	' Comments : ' Parameters : ' Returns : ' Created : 08-Mar-00 Jeff Leiner ' Modified : ' ' --
	cboSelect2_AfterUpdate	132 - 188	' Comments : ' Parameters : ' Returns : ' Created : ' Modified : 01 Jul 1998 JSL ' ' --
	cboSelect2_GotFocus	190 - 214	' Comments : ' Parameters : ' Returns : ' Created : 08-Mar-00 Jeff Leiner ' Modified : ' ' --
	cmdDelete_Click	216 - 266	' Comments : ' Parameters : ' Returns : ' Created : ' Modified : 01 Jul 1998 JSL ' ' --
	cmdEdit_Click	268 - 304	' Comments : Edit the selected record in the list box ' Parameters : ' Returns : ' Created : 10-Mar-00 Jeff Leiner ' Modified : ' ' --
	cmdNew_Click	306 - 329	' Comments : ' Parameters : ' Returns : ' Created : ' Modified : 01 Jul 1998 JSL ' ' --
	cmdSave_Click	331 - 371	' Comments : ' Parameters: - ' Modified : ' ' --
	Form_Activate	373 - 401	' Comments : ' Parameters : ' Returns : ' Created : ' Modified : 01 Jul 1998 JSL ' ' --
	Form_BeforeUpdate	403 - 422	' Comments : ' Parameters : Cancel ' Returns : ' Created : ' Modified : 01 Jul 1998 JSL ' ' --
	Form_Delete	424 - 449	' Comments : ' Parameters : Cancel ' Returns : ' Created : ' Modified : 01 Jul 1998 JSL ' ' --
	Form_Dirty	451 - 453	
	Form_Error	455 - 497	' Comments : ' Parameters : DataErr ' Response ' Returns : ' Created : ' Modified : 01 Jul 1998 JSL ' ' --
	Form_Open	499 - 579	' Comments : ' Parameters : Cancel ' Returns : ' Created : ' Modified : 01 Jul 1998 JSL ' ' --
	lstSelect_Click	581 - 610	' Comments : ' Parameters : ' Returns : ' Created : 08-Mar-00 Jeff Leiner ' Modified : ' ' --
	MarkCurrentPrice	30 - 73	' Comments : For the supplier / product Selected, Select the currently ' : Active Price in the listbox ' Parameters : ' Returns : ' Created : 10-Mar-00

ModuleName	Procedure	Lines	Comments
			Jeff Leiner ' Modified : ' ' ---
	SetText	612 - 632	' Comments : To Set each label to the correct text. ' Parameters : ' Returns : ' Created : ' Modified : 01 Jul 1998 JSL ' : 05/18/98 JSL - modified Commodity form to create Case Size ' ---
frmDataSources	(General Declarations)	1 - 32	'+ ' Module : Form_frmDataSources ' Description: Form allowing user to modify data sources for various types ' Procedures : cboSelect1_AfterUpdate() ' cboSelect1_GotFocus() ' cmdDelete_Click() ' cmdNew_Click() ' cmdSave_Click() ' Form_Activate() ' Form_BeforeInsert(Cancel As Integer) ' Form_BeforeUpdate(Cancel As Integer) ' Form_Dirty(Cancel As Integer) ' Form_Error(DataErr As Integer, Response As Integer) ' Form_Open(Cancel As Integer) ' lstSelect_Click() ' SetText() ' txtCode_BeforeUpdate(Cancel As Integer) ' txtName_AfterUpdate()
	cboSelect1_AfterUpdate	40 - 64	'+ ' Procedure : cboSelect1_AfterUpdate ' Comments : Runs after the user selects a data source type ' Parameters: - ' Modified : 03 Jun 2003 LBlanken ' 16 Jun 2003 LBlanken '-
	cboSelect1_GotFocus	74 - 93	
	cmdDelete_Click	103 - 157	
	cmdNew_Click	167 - 198	
	cmdSave_Click	207 - 258	
	Form_Activate	267 - 286	
	Form_BeforeInsert	295 - 310	
	Form_BeforeUpdate	319 - 355	
	Form_Dirty	364 - 379	
	Form_Error	389 - 423	
	Form_Open	433 - 473	
	lstSelect_Click	482 - 510	
	SetText	520 - 534	
	txtCode_BeforeUpdate	545 - 604	
	txtName_AfterUpdate	613 - 629	
frmFundingSources	(General Declarations)	1 - 33	'+ ' Module : Form_frmFundingSources ' Description: Form allowing user

ModuleName	Procedure	Lines	Comments
			to modify Funding Sources for various types ' Procedures : ' ' cmdDelete_Click() ' cmdNew_Click() ' cmdSave_Click() ' Form_Activate() ' Form_BeforeInsert(Cancel As Integer) ' Form_BeforeUpdate(Cancel As Integer) ' Form_Dirty(Cancel As Integer) ' Form_Error(DataErr As Integer, Response As Integer) ' Form_Open(Cancel As Integer) ' lstSelect_Click() ' SetText() ' txtCode_BeforeUpdate(Cancel As Integer) ' txtName_AfterUpdate()
	cmdDelete_Click	42 - 96	'+ ' Procedure : cmdDelete_Click ' Comments : Runs when the user clicks on the delete button ' Parameters: - ' Modified : 01 Jul 1998 JSL ' 03 Jun 2003 LBlanken ' 16 Jun 2003 LBlanken '-
	cmdNew_Click	106 - 137	
	cmdSave_Click	145 - 196	
	Form_Activate	205 - 224	
	Form_BeforeUpdate	233 - 269	
	Form_Dirty	278 - 293	
	Form_Error	303 - 337	
	Form_Open	347 - 379	
	lstSelect_Click	388 - 416	
	SetText	426 - 440	
	txtCode_BeforeUpdate	451 - 501	
	txtName_AfterUpdate	510 - 526	
frmGraphConsumption	(General Declarations)	1 - 33	'+ ' Module : Form_frmGraphConsumption ' Description: ' Procedures : cboDisplay_AfterUpdate() ' cboFrom_AfterUpdate() ' cboGrouping_AfterUpdate() ' cboThrough_AfterUpdate() ' cmdPreview_Click() ' cmdPrint_Click() ' DisplayGraph() ' DisplayReport(intPrint As Integer) ' Form_Activate() ' Form_Open(Cancel As Integer) ' GetReportSQL() ' ini_cboProduct() ' ini_cboYears() ' lstDisplay1_AfterUpdate() ' optDisplay_AfterUpdate() ' SetText() ' update_Graph() ' UpdateControls(lngColumn As Long)
	cboDisplay_AfterUpdate	41 - 59	'+ ' Procedure : cboDisplay_AfterUpdate ' Comments : ' Parameters: - ' Modified : 11 Jul 2003 LBlanken ' 22 Dec 2003 LBlanken '-
	cboFrom_AfterUpdate	70 - 97	
	cboGrouping_AfterUpdate	106 - 131	

ModuleName	Procedure	Lines	Comments
	cboPeriod_AfterUpdate	133 - 163	' Comments : ' Parameters : ' Returns : ' Created : ' Modified : 15 Jul 2009 mahmed ' ' --
	cboPeriod_GotFocus	166 - 168	
	cboPeriod_MouseDown	170 - 172	'Call SetDropDownWidth([cboPeriod])
	cboThrough_AfterUpdate	181 - 199	
	cmdPDF_Click	201 - 230	'+ ' Procedure : cmdPDF_Click ' Comments : ' Parameters: - ' Modified : 01 Jul 1998 JSL ' Modified : 11 Jul 2003 LBlanken '-
	cmdPreview_Click	239 - 264	
	cmdPrint_Click	273 - 297	
	cmdShowHide_Click	517 - 522	
	DisplayGraph	306 - 422	
	DisplayReport	431 - 514	
	Form_Activate	531 - 543	
	Form_Open	553 - 646	
	GetReportSQL	655 - 773	' Comments : Update the Rowsource for the Graph object ' Parameters : ' Returns : ' Created : 08/17/98 JSL ' Modified : ' ' --
	ini_cboProduct	782 - 860	' Comments : ' Parameters : ' Returns : ' Created : ' Modified : 01 Jul 1998 JSL ' ' -- ' Set up lstDisplay1 with ProductID of first record in Consumption
	ini_cboYears	869 - 914	' Comments : ' Parameters : ' Returns : ' Created : ' Modified : 01 Jul 1998 JSL ' ' -- ' Set up cboStart/End Year boxes from MonthlyStock
	lstDisplay1_AfterUpdate	922 - 1005	
	lstDisplay1_KeyDown	1007 - 1009	
	lstDisplay1_KeyUp	1011 - 1013	
	optBU_AfterUpdate	1021 - 1034	

ModuleName	Procedure	Lines	Comments
	optDisplay_AfterUpdate	1043 - 1133	' Comments : ' Parameters: - ' Modified : ' ' ---
	SetText	1141 - 1167	' Comments : ' Parameters : ' Returns : ' Created : ' Modified : 01 Jul 1998 JSL ' ' ---
	ShowFirstSelected	1285 - 1339	
	update_Graph	1175 - 1239	' Comments : ' Parameters : ' Returns : ' Created : ' Modified : 01 Jul 1998 JSL ' ' ---
	UpdateControls	1248 - 1284	
frmGraphCYP	(General Declarations)	1 - 29	'+ ' Module : Form_frmGraphCYP ' Description: ' Procedures : getSelectedProducts() ' cboFrom_AfterUpdate() ' cboGrouping_AfterUpdate() ' cboThrough_AfterUpdate() ' cmdExport_Click() ' cmdPreview_Click() ' cmdPrint_Click() ' DisplayGraph() ' DisplayReport(intPrint As Integer) ' Form_Activate() ' Form_Open(Cancel As Integer) ' ini_cboProduct() ' ini_cboYears() ' lstDisplay_AfterUpdate() ' optDisplay_AfterUpdate() ' SetText() ' update_Graph()
	cboFrom_AfterUpdate	98 - 131	' Comments : ' Parameters : ' Returns : ' Created : ' Modified : 01 Jul 1998 JSL ' ' ---
	cboGrouping_AfterUpdate	139 - 170	' Comments : ' Parameters : ' Returns : ' Created : 08/31/98 JSL ' Modified : ' ' ---
	cboPeriod_AfterUpdate	172 - 203	
	cboPeriod_GotFocus	206 - 208	
	cboThrough_AfterUpdate	216 - 240	' Comments : ' Parameters : ' Returns : ' Created : ' Modified : 01 Jul 1998 JSL ' ' ---
	CmdExport_Click	248 - 318	' Comments : Export the selected data to an Excel Spreadsheet for use ' : in an FPPMES Database. ' Parameters : ' Returns : ' Created : 10-Mar-00 Jeff Leiner ' Modified : ' ' ---
	cmdPDF_Click	320 - 349	'+ ' Procedure : cmdPDF_Click ' Comments : ' Parameters: - ' Modified : 11 Jul 2003 LBlanken '-
	cmdPreview_Click	357 - 387	' Comments : ' Parameters : ' Returns : ' Created : ' Modified : 01 Jul 1998 JSL ' ' ---

ModuleName	Procedure	Lines	Comments
	cmdPrint_Click	395 - 426	' Comments : ' Parameters : ' Returns : ' Created : ' Modified : 01 Jul 1998 JSL ' ' ---
	cmdShowHide_Click	613 - 618	
	DisplayGraph	434 - 548	' Comments : Update the Rowsource for the Graph object ' Parameters : ' Returns : ' Created : 08/17/98 JSL ' Modified : ' ' ---
	DisplayReport	556 - 610	' Comments : ' Parameters : ' Returns : ' Created : ' Modified : 01 Jul 1998 JSL ' ' ---
	Form_Activate	626 - 646	' Comments : ' Parameters : ' Returns : ' Created : ' Modified : 01 Jul 1998 JSL ' ' ---
	Form_Error	656 - 686	
	Form_Open	694 - 776	' Comments : ' Parameters : Cancel ' Returns : ' Created : ' Modified : 01 Jul 1998 JSL ' ' --- ' Set up cboProductID with ProductID of first record in Inventory
	getSelectedProducts	37 - 90	'+ ' Procedure : getSelectedProducts ' Comments : ' Parameters: - ' Returns : String - ' Modified : 11 Jul 2003 LBlanken ' -
	ini_cboProduct	785 - 851	' Comments : ' Parameters : ' Returns : ' Created : ' Modified : 01 Jul 1998 JSL ' ' --- ' Set up lstDisplay with ProductID of first record in Consumption
	ini_cboYears	860 - 905	' Comments : ' Parameters : ' Returns : ' Created : ' Modified : 01 Jul 1998 JSL ' ' --- ' Set up cboStart/End Year boxes from MonthlyStock
	lstDisplay_AfterUpdate	913 - 956	' Comments : ' Parameters : ' Returns : ' Created : ' Modified : 01 Jul 1998 JSL ' ' ---
	lstDisplay_KeyDown	958 - 960	
	lstDisplay_KeyUp	962 - 964	
	optBU_AfterUpdate	972 - 985	
	optDisplay_AfterUpdate	993 - 1071	' Comments : ' Parameters: - ' Modified : ' ' ---
	SetText	1079 - 1107	' Comments : ' Parameters : ' Returns : ' Created : ' Modified : 01 Jul 1998 JSL ' ' ---
	ShowFirstSelected	1197 -	

ModuleName	Procedure	Lines	Comments
		1251	
	update_Graph	1115 - 1195	' Comments : ' Parameters : ' Returns : ' Created : ' Modified : 01 Jul 1998 JSL ' ' --
frmGraphStockStatus	(General Declarations)	1 - 37	'+ ' Module : Form_frmGraphStockStatus ' Description: ' Procedures : SetListboxSelected(lstBox As ListBox, fSelected As Boolean) ' cboDisplay_AfterUpdate() ' cboFrom_AfterUpdate() ' cboGrouping_AfterUpdate() ' cboThrough_AfterUpdate() ' chkAllSupplier_AfterUpdate() ' cmdPreview_Click() ' cmdPrint_Click() ' DisplayGraph() ' DisplayReport(intPrint As Integer) ' Form_Activate() ' Form_Open(Cancel As Integer) ' GetReportSQL() ' ini_cboProduct() ' ini_cboYears() ' lstDisplay1_AfterUpdate() ' lstDisplay2_AfterUpdate() ' optDisplay_AfterUpdate() ' SetText() ' update_Graph() ' UpdateControls(lngColumn As Long)
	cboDisplay_AfterUpdate	80 - 98	
	cboFrom_AfterUpdate	106 - 142	' Comments : ' Parameters : ' Returns : ' Created : ' Modified : 01 Jul 1998 JSL ' Modified : 14 July 2009 mahmed ' set and display user selected date range ' --
	cboGrouping_AfterUpdate	150 - 181	' Comments : ' Parameters : ' Returns : ' Created : 08/31/98 JSL ' Modified : ' ' --
	cboPeriod_AfterUpdate	183 - 213	' Comments : ' Parameters : ' Returns : ' Created : ' Modified : 15 Jul 2009 mahmed ' ' --
	cboPeriod_GotFocus	216 - 218	
	cboThrough_AfterUpdate	226 - 252	' Comments : ' Parameters : ' Returns : ' Created : ' Modified : 01 Jul 1998 JSL ' ' --
	chkAllSupplier_AfterUpdate	260 - 288	' Comments : ' Parameters: - ' Modified : ' ' --
	cmdPDF_Click	290 - 321	'+ ' Procedure : cmdPDF_Click ' Comments : ' Parameters: - ' Modified : 11 Jul 2003 LBlanken '-
	cmdPreview_Click	329 - 362	' Comments : Open the Status Graph Report (in Preview mode) ' Parameters : ' Returns : ' Created : ' Modified : 01 Jul 1998 JSL ' --
	cmdPrint_Click	370 - 402	' Comments : Open the Status Graph Report (in Preview mode) ' Parameters : ' Returns : ' Created : ' Modified : 01 Jul 1998 JSL ' ' --

ModuleName	Procedure	Lines	Comments
	cmdShowHide_Click	602 - 607	
	DisplayGraph	410 - 518	' Comments : Update the Rowsource for the Graph object ' Parameters : ' Returns : ' Created : 08/17/98 JSL ' Modified : ' ' ---
	DisplayReport	526 - 599	' Comments : Open the Status Graph Report (in Preview mode) ' Parameters : ' Returns : ' Created : ' Modified : 01 Jul 1998 JSL ' ' ---
	Form_Activate	615 - 633	' Comments : Reset the text. ' Parameters : ' Returns : ' Created : ' Modified : 01 Jul 1998 JSL ' ' ---
	Form_Error	643 - 673	
	Form_Open	683 - 783	
	GetReportSQL	792 - 890	' Comments : Update the Rowsource for the Graph object ' Parameters : ' Returns : ' Created : 08/17/98 JSL ' Modified : ' ' ---
	ini_cboProduct	899 - 980	' Comments : ' Parameters : ' Returns : ' Created : ' Modified : 01 Jul 1998 JSL ' ' --- ' Set up lstDisplay1 with ProductID of first record in Consumption
	ini_cboYears	989 - 1040	' Comments : ' Parameters : ' Returns : ' Created : ' Modified : 01 Jul 1998 JSL ' ' --- ' Set up cboStart/End Year boxes from MonthlyStock
	lstDisplay1_AfterUpdate	1048 - 1136	
	lstDisplay1_KeyDown	1138 - 1140	
	lstDisplay1_KeyUp	1142 - 1144	
	lstDisplay2_AfterUpdate	1152 - 1176	' Comments : ' Parameters: - ' Modified : ' ' ---
	lstDisplay2_KeyDown	1182 - 1184	
	lstDisplay2_KeyUp	1186 - 1188	
	optBU_AfterUpdate	1178 -	

ModuleName	Procedure	Lines	Comments
		1180	
	optDisplay_AfterUpdate	1197 - 1282	
	SetListboxSelected	45 - 71	'+ ' Procedure : SetListboxSelected ' Comments : ' Parameters: lstBox ' fSelected - ' Modified : 11 Jul 2003 LBlanken '-
	SetText	1290 - 1317	' Comments : ' Parameters : ' Returns : ' Created : ' Modified : 01 Jul 1998 JSL ' ' --
	ShowFirstSelected	1427 - 1481	
	update_Graph	1325 - 1379	' Comments : ' Parameters : ' Returns : ' Created : ' Modified : 01 Jul 1998 JSL ' ' --
	UpdateControls	1388 - 1425	
frmInvCount	(General Declarations)	1 - 30	' Module : Form_frmInvCount ' Description: ' Procedures : FindClosestRecord() ' cboSelect2_AfterUpdate() ' cmdDelete_Click() ' cmdNew_Click() ' cmdSave_Click() ' Form_Activate() ' Form_BeforeInsert(Cancel As Integer) ' Form_BeforeUpdate(Cancel As Integer) ' Form_Error(DataErr As Integer, Response As Integer) ' Form_Open(Cancel As Integer) ' lstSelect_Click() ' SetText() ' txtDate_AfterUpdate() ' txtDate_BeforeUpdate(Cancel As Integer)
	cboSelect2_AfterUpdate	90 - 102	' Comments : ' Parameters: - ' Modified : ' ' --
	cboSelect2_GotFocus	104 - 127	' Comments : ' Parameters : ' Returns : ' Created : 08-Mar-00 Jeff Leiner ' Modified : ' ' --
	cmdDelete_Click	129 - 183	' Comments : ' Parameters : ' Returns : ' Created : ' Modified : 01 Jul 1998 JSL ' ' --
	cmdNew_Click	185 - 211	' Comments : Open the Detail Form to A new record ' Parameters : ' Returns : ' Created : 07-Feb-00 Jeff Leiner ' Modified : ' ' --
	cmdSave_Click	213 - 250	' Comments : ' Parameters: - ' Modified : ' ' --
	FindClosestRecord	31 - 88	' Comments : ' Parameters : ' Returns : ' Created : 04-Feb-00 Jeff Leiner ' Modified : ' ' --

ModuleName	Procedure	Lines	Comments
	Form_Activate	252 - 261	' Comments : ' Parameters: - ' Modified : ' ' --
	Form_BeforeInsert	263 - 278	' Comments : ' Parameters: Cancel - ' Modified : ' ' --
	Form_BeforeUpdate	280 - 329	' Comments : ' Parameters: Cancel - ' Modified : ' ' --
	Form_Dirty	331 - 333	
	Form_Error	335 - 387	' Comments : Handles validation Errors ' Parameters : DataErr ' Response ' Returns : ' Created : ' Modified : 01 Jul 1998 JSL ' ' --
	Form_Open	389 - 519	' Comments : ' Parameters : cancel ' Returns : ' Created : ' Modified : 01 Jul 1998 JSL ' ' -- ' Set up cboSelect2 with ProductID of first record in Consumption
	lstSelect_Click	521 - 536	' Comments : ' Parameters: - ' Modified : ' ' --
	SetText	538 - 557	' Comments : ' Parameters : ' Returns : ' Created : ' Modified : 01 Jul 1998 JSL ' ' --
	txtDate_AfterUpdate	559 - 613	' Comments : ' Parameters : ' Returns : ' Created : ' Modified : 01 Jul 1998 JSL ' ' --
	txtDate_BeforeUpdate	615 - 642	' Comments : See if a record exists for the txtDate already ' Parameters : ' Returns : ' Created : 07/13/98 JSL ' Modified : ' ' --
frmMain	(General Declarations)	1 - 22	'+ ' Module : Form_frmMain ' Description: ' Procedures : cboLanguage_AfterUpdate() ' Form_Open(Cancel As Integer) ' SetText() ' Treeview1_GotFocus() ' Treeview1_LostFocus() ' Treeview1_NodeClick(ByVal Node As Object)
	cboLanguage_AfterUpdate	30 - 48	'+ ' Procedure : cboLanguage_AfterUpdate ' Comments : Updates language based on value ' Parameters: - ' Modified : 01 Jul 1998 JSL ' 16 Jun 2003 LBlanken '-
	Form_Activate	60 - 90	
	Form_Close	92 - 95	
	Form_Open	104 - 153	
	RequeryConsumption	344 - 346	

ModuleName	Procedure	Lines	Comments
	SetText	161 - 179	
	Treeview1_GotFocus	187 - 202	
	Treeview1_LostFocus	210 - 224	
	Treeview1_NodeClick	233 - 339	
	UpdateLanguage	340 - 342	
frmMain_temp	(General Declarations)	1 - 22	'+ ' Module : Form_frmMain ' Description: ' Procedures : cboLanguage_AfterUpdate() ' Form_Open(Cancel As Integer) ' SetText() ' Treeview1_GotFocus() ' Treeview1_LostFocus() ' Treeview1_NodeClick(ByVal Node As Object)
	cboLanguage_AfterUpdate	30 - 48	'+ ' Procedure : cboLanguage_AfterUpdate ' Comments : Updates language based on value ' Parameters: - ' Modified : 01 Jul 1998 JSL ' 16 Jun 2003 LBlanken '-
	Form_Close	53 - 56	
	Form_Open	65 - 117	
	SetText	125 - 143	
	Treeview1_GotFocus	151 - 166	
	Treeview1_LostFocus	174 - 188	
	Treeview1_NodeClick	197 - 298	
	UpdateLanguage	299 - 301	
frmProducts	(General Declarations)	1 - 26	' Module : Form_frmProducts ' Description: Allows users to maintain the list of products used ' throughout the system. ' Procedures : CheckMinMaxs() ' cmdCategories_Click() ' cmdDelete_Click() ' cmdNew_Click() ' cmdSave_Click() ' Form_Activate() ' Form_AfterUpdate() ' Form_BeforeInsert(Cancel As Integer) ' Form_BeforeUpdate(Cancel As Integer) ' Form_Dirty(Cancel As Integer) ' Form_Open(Cancel As Integer) ' Form_Unload(Cancel As Integer) ' lstSelect_Click() ' SetText() ' txtName_AfterUpdate() ' Modified : 06/03/03 lkb ' 05/22/03 lkb Cleaned with Total Visual CodeTools ' ---
	checkDEOYS	27 - 50	' Comments : Compare the Program Min and Max to the DEOYS ' : and make sure DEOYS is within the two. ' Parameters : - ' Returns : Boolean ' Created : 13-Jun-06 mtracy ' Modified : ' --

ModuleName	Procedure	Lines	Comments
	CheckMinMaxs	52 - 144	' Comments : Compare the Product Mins and Maxs and the Central level ' : Mins and maxs and make sure they all fit in the proper ' : intervals. ' Parameters : - ' Returns : Boolean ' Created : 25-Feb-00 Jeff Leiner ' Modified : ' --
	cmdCategories_Click	146 - 153	' Comments : opens the assign category form ' Parameters: - ' Modified : 06/03/03 lkb ' -- 'open popup form
	cmdDelete_Click	155 - 224	' Comments : deletes selected item in listbox ' Parameters : - ' Modified : 07/01/98 JSL ' 06/03/03 lkb ' --
	cmdImport_Click	226 - 229	
	cmdNew_Click	231 - 266	' Comments : Open the Detail Form to A new record ' Parameters : - ' Returns : - ' Created : 07-Feb-00 Jeff Leiner ' Modified : 06/03/03 lkb ' --
	cmdSave_Click	268 - 327	' Comments : saves the recod and updates listbox/form ' Parameters: - ' Modified : ' --
	Form_Activate	329 - 350	' Comments : sets value of list box ' Parameters: - ' Modified : 06/03/03 lkb ' --
	Form_AfterUpdate	352 - 363	' Comments : sets global product id ' Parameters: - ' Modified : 06/03/03 lkb ' --
	Form_BeforeInsert	365 - 391	' Comments : Find the Default Supplier ' Parameters : - ' Returns : - ' Created : 07/13/98 JSL ' Modified : ' --
	Form_BeforeUpdate	393 - 447	' Comments : validates data ' Parameters: Cancel - exit process ' Modified : 06/03/03 lkb ' --
	Form_Dirty	449 - 455	' Comments : sets global dirty flag ' Parameters: Cancel - exit process ' Modified : 06/03/03 lkb ' --
	Form_Error	465 - 510	
	Form_Open	512 - 575	' Comments : sets rowsource for listbox, set value of listbox ' to global product ID (if global is null, sets it) ' Parameters : cancel - exit process ' Returns : - ' Modified : 07/01/98 JSL ' 06/03/03 lkb ' --
	Form_Unload	577 - 619	' Comments : if the current product isn't active, Check to see ' : if there is at least one active product ' Parameters : cancel - exit process ' Returns : - ' Created : 18-Feb-00 Jeff Leiner ' Modified : ' --

ModuleName	Procedure	Lines	Comments
	lstSelect_Click	621 - 650	' Comments : Prompt user to save form if dirty, else requery form ' Parameters: - ' Modified : 06/03/03 lkb ' --
	SetText	652 - 674	' Comments : sets the captions, statusbar, styles for controls on form ' Parameters : - ' Modified : 07/01/98 JSL ' 06/03/03 lkb ' --
	txtDesCMax_AfterUpdate	676 - 680	'this is causing too many validations and making it difficult to add data 'therefore I'm moving it to run on save. (LKB 4/4/2007) 'CheckMinMaxs Me!txtDesCMax, False
	txtDesCMin_AfterUpdate	682 - 686	'this is causing too many validations and making it difficult to add data 'therefore I'm moving it to run on save. (LKB 4/4/2007) 'CheckMinMaxs Me!txtDesCMin, False
	txtDesMax_AfterUpdate	688 - 693	
	txtDesMin_AfterUpdate	695 - 700	
	txtDesStock_AfterUpdate	702 - 708	
	txtName_AfterUpdate	710 - 721	' Comments : generate codes after name is entered ' Parameters: - ' Modified : 06/03/03 lkb ' --
	UpdateControls	723 - 758	
frmProgram	(General Declarations)	1 - 53	'+ ' Module : Form_frmProgram ' Description: Manages program form ' Procedures : IsImportVisible() ' SetFormMode() ' cboDisplay_AfterUpdate() ' cboDisplay_GotFocus() ' cmdAdd_Click() ' cmdBackup_Click() ' cmdCPTExport_Click() ' cmdDelete_Click() ' cmdImport_Click() ' cmdImport_GotFocus() ' cmdImportShipmentData_Click() ' cmdImportShipmentData_GotFocus() ' cmdRestore_Click() ' DirtyContinue() ' edit_error() ' Form_AfterInsert() ' Form_AfterUpdate() ' Form_BeforeUpdate(Cancel As Integer) ' Form_Current() ' Form_Delete(Cancel As Integer) ' Form_Dirty(Cancel As Integer) ' Form_Error(DataErr As Integer, Response As Integer) ' Form_Open(Cancel As Integer) ' Form_Unload(Cancel As Integer) ' LockFields(intLock As Integer) ' MakeActive(fActive As Boolean, strCode As String) ' MakeCurrent() ' OpenDialog(strFrmName As String) ' ResetStatus() ' SetText() ' txtBlocker_Enter() ' UpdateBackEnd()
	cboCountryCode_AfterUpdate	54 - 67	
	cboCountryCode_NotInList	69 - 74	'mtracy 4/20/07 for bugzilla 14546

ModuleName	Procedure	Lines	Comments
	cboDisplay_GotFocus	82 - 97	
	cboLanguage_AfterUpdate	100 - 102	
	cboLanguage_GotFocus	104 - 106	
	cboLanguage_NotInList	108 - 114	'LBLANKEN 4/20/07 for bugzilla 14546
	cmdAdd_Click	124 - 207	
	cmdBackup_Click	217 - 262	
	edit_error	272 - 339	
	Form_AfterInsert	348 - 374	
	Form_AfterUpdate	383 - 416	
	Form_BeforeUpdate	425 - 532	
	Form_Close	535 - 545	
	Form_Current	554 - 576	
	Form_Dirty	584 - 598	
	Form_Error	608 - 666	
	Form_Open	675 - 761	
	Form_Unload	770 - 793	
	LockFields	802 - 842	
	OpenDialog	851 - 863	
	ResetStatus	872 - 884	
	txtBlocker_Enter	893 - 906	
	txtCode_AfterUpdate	908 - 912	
frmRptAnnualCosts	(General Declarations)	1 - 23	'+ ' Module : Form_frmRptAnnualCosts ' Description: Code for Annual Cost Report parameter form ' Procedures : cboFrom_AfterUpdate() ' cboThrough_AfterUpdate() ' cmdPreview_Click() ' cmdPrint_Click() ' FillCombos(ctlCombo As Control, intMedian As Integer, intStart As Integer, intEnd As Integer) ' FillFilterCombos(strType As String) ' Form_Load() ' Form_Open(Cancel As Integer) ' MakeDataSet() ' SetText()
	cboDisplay_AfterUpdate	24 - 69	
	cboFrom_AfterUpdate	78 - 92	

ModuleName	Procedure	Lines	Comments
	cboPeriod_AfterUpdate	95 - 110	
	cboPeriod_GotFocus	113 - 115	
	cboThrough_AfterUpdate	124 - 140	
	cmdPDF_Click	142 - 174	'+ ' Procedure : cmdPreview_Click ' Comments : diplay report in preview mode ' Parameters: - ' Modified : 01 Jul 1998 JSL ' 11 Jul 2003 LBlanken '-
	cmdPreview_Click	183 - 208	
	cmdPrint_Click	217 - 236	
	FillCombos	249 - 296	
	FillFilterCombos	306 - 421	
	Form_Error	431 - 461	
	Form_Load	470 - 498	
	Form_Open	507 - 543	
	lstDisplay1_AfterUpdate	672 - 701	
	lstDisplay2_AfterUpdate	703 - 726	
	MakeDataSet	553 - 642	
	SetText	651 - 670	
frmRptPipelineAction	(General Declarations)	1 - 19	'+ ' Module : Form_frmRptPipelineAction ' Description: Code for Pipeline Action Report parameter form ' Procedures : cmdPreview_Click() ' cmdPrint_Click() ' cmdShowHide_Click() ' DisplayReport(intView As Integer) ' Form_Open(Cancel As Integer) ' SetText()
	cboDisplay_AfterUpdate	20 - 42	
	cboFrom_AfterUpdate	50 - 75	
	cboPeriod_AfterUpdate	76 - 98	
	cboPeriod_GotFocus	101 - 103	
	cboThrough_AfterUpdate	111 - 127	
	cmdPDF_Click	129 - 148	'+ ' Procedure : cmdPDF_Click ' Comments : display report in preview mode ' Parameters: - ' Modified : 01 Jul 1998 JSL ' 11 Jul 2003 LBlanken '-
	cmdPreview_Click	157 - 170	

ModuleName	Procedure	Lines	Comments
	cmdPrint_Click	179 - 192	
	cmdShowHide_Click	200 - 229	
	DisplayReport	238 - 314	
	Form_Error	324 - 354	
	Form_Open	362 - 401	
	SetText	410 - 428	
	VerifyDateRange	438 - 477	
frmRptPipelineProblem	(General Declarations)	1 - 20	'+ ' Module : Form_frmRptPipelineProblem ' Description: Code for Pipeline Problem Report parameter form ' Procedures : cmdPreview_Click() ' cmdPrint_Click() ' DisplayReport(intView As Integer) ' Form_Open(Cancel As Integer) ' SetText()
	cboDisplay_AfterUpdate	21 - 32	
	cmdPDF_Click	34 - 55	'+ ' Procedure : cmdPDF_Click ' Comments : display report in preview mode ' Parameters: - ' Modified : 01 Jul 1998 JSL ' 11 Jul 2003 LBlanken '-
	cmdPreview_Click	64 - 79	
	cmdPrint_Click	88 - 103	
	cmdShowHide_Click	172 - 203	
	DisplayReport	112 - 164	
	Form_Open	212 - 234	
	getEndYear	262 - 295	' Comments : ' Parameters : ' Returns : ' Created : ' Modified : 01 Jul 1998 JSL ' ' --
	SetText	243 - 260	
frmRptProcurementTable	(General Declarations)	1 - 31	'+ ' Module : Form_frmRptProcurementTable ' Description: Code for Procurement Report parameter form ' Procedures : cboDisplay_AfterUpdate() ' cboUseMonths_AfterUpdate() ' cboYear_AfterUpdate() ' cmdPreview_Click() ' cmdPrint_Click() ' cmdShowHide_Click() ' DisplayReport(intView As Integer) ' Form_Load() ' Form_Open(Cancel As Integer) ' getValidMonths(strID As String) ' ini_cboProduct() ' ini_cboYear() ' lstDisplay1_AfterUpdate() ' SetMinMax() ' SetText() ' UpdateControls(lngColumn As Long)
	cboDisplay_AfterUpdate	38 - 100	'+ ' Procedure : cboDisplay_AfterUpdate ' Comments : updates listbox

ModuleName	Procedure	Lines	Comments
			based on category selected ' Parameters: - ' Modified : 11 Jul 2003 LBlanken '-
	cboUseMonths_AfterUpdate	108 - 124	
	cboYear_AfterUpdate	133 - 162	
	cmdPDF_Click	164 - 192	' Procedure : cmdPDF_Click ' Comments : display report in preview mode ' Parameters: - ' Modified : 1 Aug 2006 LBlanken '-
	cmdPreview_Click	201 - 225	
	cmdPrint_Click	233 - 254	
	cmdShowHide_Click	262 - 304	
	DisplayReport	313 - 369	
	Form_Error	379 - 409	
	Form_Load	418 - 431	
	Form_Open	440 - 531	
	getValidMonths	539 - 593	
	ini_cboProduct	603 - 658	
	ini_cboYear	668 - 709	
	lstDisplay1_AfterUpdate	719 - 816	
	lstDisplay1_KeyDown	951 - 953	
	lstDisplay1_KeyUp	955 - 957	
	SetMinMax	824 - 861	
	SetText	870 - 889	
	ShowDOM	940 - 949	
	ShowFirstSelected	959 - 1000	
	UpdateControls	898 - 939	
frmRptShipmentOrders	(General Declarations)	1 - 35	'+ ' Module : Form_frmRptShipmentOrders ' Description: Code for Shipment Order Report parameter form ' Procedures : cboFrom_AfterUpdate() ' cboThrough_AfterUpdate() ' cmdPreview_Click() ' cmdPrint_Click() ' cmdShowHide_Click() ' CreateShipQuery() ' DisplayReport(lngMode As Long) ' FillFilterCombos(strType As String) ' Form_Load() '

ModuleName	Procedure	Lines	Comments
			lstDisplay1_AfterUpdate() ' lstDisplay2_AfterUpdate() ' MakeFilters() ' SetListSource() ' SetPageBreak(strReport As String, intMode As Integer) ' SetText() ' VerifyDateRange()
	cboDisplay_AfterUpdate	36 - 79	
	cboFrom_AfterUpdate	88 - 112	
	cboPeriod_AfterUpdate	114 - 137	
	cboPeriod_GotFocus	141 - 143	
	cboThrough_AfterUpdate	152 - 170	
	cmdPDF_Click	172 - 202	'+ ' Procedure : cmdPDF_Click ' Comments : display report in preview mode ' Parameters: - ' Modified : 1 AUG 2006 LBlanken '-
	cmdPreview_Click	211 - 237	
	cmdPrint_Click	246 - 272	
	cmdShowHide_Click	280 - 323	
	CreateShipQuery	334 - 372	
	DisplayReport	381 - 434	
	FillFilterCombos	444 - 507	
	Form_Error	517 - 547	
	Form_Load	558 - 603	
	Form_Open	611 - 649	
	lstDisplay1_AfterUpdate	658 - 691	
	lstDisplay1_KeyDown	693 - 695	
	lstDisplay1_KeyUp	697 - 699	
	lstDisplay2_AfterUpdate	709 - 741	
	lstDisplay2_KeyDown	1023 - 1025	
	lstDisplay2_KeyUp	1027 - 1029	
	MakeFilters	751 - 846	
	SetListSource	855 - 893	
	SetPageBreak	903 - 932	

ModuleName	Procedure	Lines	Comments
	SetText	941 - 966	
	VerifyDateRange	976 - 1021	
frmRptShipmentSummary	(General Declarations)	1 - 36	'+ ' Module : Form_frmRptShipmentSummary ' Description: code for shipment summary report parameter form ' Procedures : cboFrom_AfterUpdate() ' cboThrough_AfterUpdate() ' cmdPreview_Click() ' cmdPrint_Click() ' cmdShowHide_Click() ' CreateShipQuery() ' DisplayReport(lngMode As Long) ' FillFilterCombos(strType As String) ' Form_Load() ' Form_Open(Cancel As Integer) ' lstDisplay1_AfterUpdate() ' lstDisplay2_AfterUpdate() ' optDisplay_AfterUpdate() ' SetListSource() ' SetPageBreak(strReport As String, intMode As Integer) ' SetText() ' VerifyDateRange()
	cboDisplay_AfterUpdate	37 - 93	
	cboFrom_AfterUpdate	102 - 141	
	cboPeriod_AfterUpdate	143 - 167	
	cboPeriod_GotFocus	170 - 172	
	cboThrough_AfterUpdate	181 - 199	
	cmdPDF_Click	201 - 233	'+ ' Procedure : cmdPDF_Click ' Comments : display report in normal mode (i.e. send to printer) ' Parameters: - ' Modified : 01 AUG 2006 LKB '-
	cmdPreview_Click	242 - 270	
	cmdPrint_Click	279 - 306	
	cmdShowHide_Click	314 - 361	
	CreateShipQuery	372 - 411	
	DisplayReport	420 - 471	
	FillFilterCombos	481 - 542	
	Form_Error	552 - 582	
	Form_Load	593 - 655	
	Form_Open	666 - 714	
	lstDisplay1_AfterUpdate	723 - 772	
	lstDisplay1_KeyDown	774 - 776	

ModuleName	Procedure	Lines	Comments
	lstDisplay1_KeyUp	778 - 780	
	lstDisplay2_AfterUpdate	789 - 821	
	lstDisplay2_KeyDown	823 - 825	
	lstDisplay2_KeyUp	827 - 829	
	MakeFilters	1046 - 1147	
	optDisplay_AfterUpdate	837 - 856	
	SetListSource	865 - 902	
	SetPageBreak	912 - 941	
	SetText	950 - 981	
	VerifyDateRange	991 - 1036	
frmRptStockStatus	(General Declarations)	1 - 36	'+ ' Module : Form_frmRptStockStatus ' Description: code for stock status report parameter form ' Procedures : SetListboxSelected(lstBox As ListBox, fSelected As Boolean) ' cboDisplay_AfterUpdate() ' cboFrom_AfterUpdate() ' cboGrouping_AfterUpdate() ' cboThrough_AfterUpdate() ' chkAllSupplier_AfterUpdate() ' cmdPreview_Click() ' cmdPrint_Click() ' cmdShowHide_Click() ' DisplayReport(intView As Integer) ' Form_Load() ' Form_Open(Cancel As Integer) ' ini_cboProduct() ' ini_cboYears() ' lstDisplay1_AfterUpdate() ' lstDisplay2_AfterUpdate() ' optDisplay_AfterUpdate() ' SetText() ' Update_Report() ' UpdateControls(lngColumn As Long)
	cboDisplay_AfterUpdate	73 - 92	
	cboFrom_AfterUpdate	103 - 130	
	cboGrouping_AfterUpdate	139 - 165	
	cboPeriod_AfterUpdate	166 - 188	
	cboPeriod_GotFocus	191 - 193	
	cboThrough_AfterUpdate	202 - 228	
	chkAllSupplier_AfterUpdate	236 - 266	
	cmdAMC_Click	268 - 280	
	cmdPDF_Click	282 - 309	'+ ' Procedure : cmdPDF_Click ' Comments : display report as PDF '

ModuleName	Procedure	Lines	Comments
			Parameters: - ' Modified : 01 AUG 2006 LKB '-
	cmdPreview_Click	318 - 343	
	cmdPrint_Click	352 - 374	
	cmdShowHide_Click	384 - 411	
	DisplayReport	422 - 607	
	Form_Error	621 - 651	
	Form_Load	660 - 676	
	Form_Open	685 - 804	
	GetSourceObject	813 - 853	
	ini_cboProduct	863 - 919	
	ini_cboYears	930 - 1012	
	lstDisplay1_AfterUpdate	1021 - 1247	
	lstDisplay1_KeyDown	1250 - 1252	
	lstDisplay1_KeyUp	1254 - 1256	
	lstDisplay2_AfterUpdate	1264 - 1283	
	lstDisplay2_KeyDown	1285 - 1287	
	lstDisplay2_KeyUp	1289 - 1291	
	optDisplay_AfterUpdate	1299 - 1436	
	setFormForMatrixReport	1579 - 1597	
	SetListboxSelected	44 - 65	'+ ' Procedure : SetListboxSelected ' Comments : sets all items in listbox to true or false ' Parameters: lstBox - listbox to update ' fSelected - true to select all, false to deselect ' Modified : 11 Jul 2003 LBlanken '-

ModuleName	Procedure	Lines	Comments
	SetText	1445 - 1464	
	ShowFirstSelected	1516 - 1570	
	UpdateControls	1473 - 1514	
frmRptStockStatusDateRange	(General Declarations)	1 - 35	'+ ' Module : Form_frmRptStockStatus ' Description: code for stock status report parameter form ' Procedures : SetListboxSelected(lstBox As ListBox, fSelected As Boolean) ' cboDisplay_AfterUpdate() ' cboFrom_AfterUpdate() ' cboGrouping_AfterUpdate() ' cboThrough_AfterUpdate() ' chkAllSupplier_AfterUpdate() ' cmdPreview_Click() ' cmdPrint_Click() ' cmdShowHide_Click() ' DisplayReport(intView As Integer) ' Form_Load() ' Form_Open(Cancel As Integer) ' ini_cboProduct() ' ini_cboYears() ' lstDisplay1_AfterUpdate() ' lstDisplay2_AfterUpdate() ' optDisplay_AfterUpdate() ' SetText() ' Update_Report() ' UpdateControls(lngColumn As Long)
	cboDisplay_AfterUpdate	72 - 91	
	cboFrom_AfterUpdate	100 - 127	
	cboGrouping_AfterUpdate	136 - 162	
	cboThrough_AfterUpdate	171 - 197	
	chkAllSupplier_AfterUpdate	205 - 235	
	cmdAMC_Click	237 - 244	
	cmdPDF_Click	246 - 273	'+ ' Procedure : cmdPDF_Click ' Comments : display report as PDF ' Parameters: - ' Modified : 01 AUG 2006 LKB '-
	cmdPreview_Click	282 - 305	
	cmdPrint_Click	314 - 336	
	cmdShowHide_Click	344 - 367	
	DisplayReport	376 - 474	
	Form_Error	484 - 514	
	Form_Load	523 - 539	
	Form_Open	548 - 657	
	GetSourceObject	666 - 698	

ModuleName	Procedure	Lines	Comments
	ini_cboProduct	708 - 764	
	ini_cboYears	775 - 857	
	lstDisplay1_AfterUpdate	866 - 1014	
	lstDisplay1_KeyDown	1017 - 1019	
	lstDisplay1_KeyUp	1021 - 1023	
	lstDisplay2_AfterUpdate	1031 - 1050	
	lstDisplay2_KeyDown	1052 - 1054	
	lstDisplay2_KeyUp	1056 - 1058	
	optDisplay_AfterUpdate	1066 - 1203	
	SetListboxSelected	43 - 64	'+ ' Procedure : SetListboxSelected ' Comments : sets all items in listbox to true or false ' Parameters: lstBox - listbox to update ' fSelected - true to select all, false to deselect ' Modified : 11 Jul 2003 LBlanken '-
	SetText	1212 - 1231	
	ShowFirstSelected	1283 - 1337	
	UpdateControls	1240 - 1281	
frmRptSupplyPlan	(General Declarations)	1 - 35	'+ ' Module : Form_frmRptStockStatus ' Description: code for stock status report parameter form ' Procedures : SetListboxSelected(lstBox As ListBox, fSelected As Boolean) ' cboDisplay_AfterUpdate() ' cboFrom_AfterUpdate() ' cboGrouping_AfterUpdate() ' cboThrough_AfterUpdate() ' chkAllSupplier_AfterUpdate() ' cmdPreview_Click() ' cmdPrint_Click() ' cmdShowHide_Click() ' DisplayReport(intView As Integer) ' Form_Load() ' Form_Open(Cancel As Integer) ' ini_cboProduct() ' ini_cboYears() ' lstDisplay1_AfterUpdate() ' lstDisplay2_AfterUpdate() ' optDisplay_AfterUpdate() ' SetText() '

ModuleName	Procedure	Lines	Comments
			Update_Report() ' UpdateControls(lngColumn As Long)
	cboDisplay_AfterUpdate	72 - 91	
	cboFrom_AfterUpdate	100 - 128	
	cboGrouping_AfterUpdate	137 - 163	
	cboPeriod_AfterUpdate	165 - 188	
	cboPeriod_GotFocus	191 - 193	
	cboThrough_AfterUpdate	202 - 228	
	chkAllSupplier_AfterUpdate	236 - 266	
	chkRoundUp_AfterUpdate	268 - 313	
	cmdPDF_Click	318 - 345	'+ ' Procedure : cmdPDF_Click ' Comments : display report as PDF ' Parameters: - ' Modified : 01 AUG 2006 LKB '-
	cmdPreview_Click	354 - 377	
	cmdPrint_Click	386 - 408	
	cmdShowHide_Click	416 - 439	
	DisplayReport	448 - 509	
	Form_Error	519 - 549	
	Form_Load	558 - 574	
	Form_Open	583 - 695	
	GetSourceObject	704 - 721	
	ini_cboProduct	731 - 786	
	ini_cboYears	797 - 871	
	lstDisplay1_AfterUpdate	880 - 1008	
	lstDisplay1_KeyDown	1011 - 1013	
	lstDisplay1_KeyUp	1015 - 1017	
	lstDisplay2_AfterUpdate	1025 - 1044	
	lstDisplay2_KeyDown	1046 -	

ModuleName	Procedure	Lines	Comments
		1048	
	lstDisplay2_KeyUp	1050 - 1052	
	optDisplay_AfterUpdate	1060 - 1146	
	SetListboxSelected	43 - 64	'+ ' Procedure : SetListboxSelected ' Comments : sets all items in listbox to true or false ' Parameters: lstBox - listbox to update ' fSelected - true to select all, false to deselect ' Modified : 11 Jul 2003 LBlanken '-
	SetText	1155 - 1174	
	ShowFirstSelected	1226 - 1272	
	UpdateControls	1183 - 1224	
frmShipments	(General Declarations)	1 - 52	'+ ' Module : Form_frmShipments ' Description: Allows user to input shipments. When form ' opens, it finds the record in the list box that is closest ' to the current date. The user then has the option to edit ' delete, or add a record. ' Procedures : MovetoClosestShipment() ' CalculateUnitCost() ' cboSelect1_AfterUpdate() ' cboSelect1_GotFocus() ' cboSelect2_AfterUpdate() ' cboSelect2_GotFocus() ' cboSupplier_AfterUpdate() ' cmdCopy_Click() ' cmdDelete_Click() ' cmdNew_Click() ' cmdSave_Click() ' Date_Consist() ' Estimate_LeadTimes(iEvent As Integer) ' Form_Activate() ' Form_AfterInsert() ' Form_BeforeInsert(Cancel As Integer) ' Form_BeforeUpdate(Cancel As Integer) ' Form_Delete(Cancel As Integer) ' Form_Dirty(Cancel As Integer) ' Form_Error(DataErr As Integer, Response As Integer) ' Form_Open(Cancel As Integer) ' GetCaseLot() ' getEstFreightCost() ' HasValidPricing() ' lstSelect_Click() ' SetText() ' txtActOrdered_AfterUpdate() ' txtActOrdered_BeforeUpdate(Cancel As Integer) ' txtActShipped_AfterUpdate() ' txtActShipped_BeforeUpdate(Cancel As Integer) ' txtDate_AfterUpdate() ' txtDate_BeforeUpdate(Cancel As Integer) ' txtQuantity_AfterUpdate()
	CalculateUnitCost	136 - 212	
	cboSelect1_AfterUpdate	220 - 270	
	cboSelect1_GotFocus	278 - 297	

ModuleName	Procedure	Lines	Comments
	cboSelect2_AfterUpdate	306 - 343	
	cboSelect2_GotFocus	353 - 372	
	cboStatus_AfterUpdate	374 - 417	
	cboSupplier_AfterUpdate	425 - 441	
	cmdBack_Click	444 - 447	
	cmdCopy_Click	457 - 501	
	cmdDelete_Click	510 - 576	
	cmdNew_Click	585 - 618	
	cmdPlan_Click	621 - 646	
	cmdSave_Click	654 - 702	
	cmdSearch_Click	929 - 936	
	Date_Consist	712 - 824	
	Estimate_LeadTimes	833 - 927	
	Form_Activate	945 - 958	
	Form_BeforeInsert	966 - 1014	
	Form_BeforeUpdate	1023 - 1090	
	Form_Dirty	1099 - 1114	
	Form_Error	1124 - 1168	
	Form_Open	1177 - 1334	
	GetCaseLot	1757 - 1854	
	getEstFreightCost	1345 - 1394	
	HasValidPricing	1407 - 1462	

ModuleName	Procedure	Lines	Comments
	lstSelect_Click	1470 - 1497	
	MovetoClosestShipment	60 - 126	'+ ' Procedure : MovetoClosestShipment ' Comments : Find the record closest to todays date. ' Parameters: - ' Created : 03 Feb 2000 Jeff Leiner ' Modified : 23 Jun 2003 LBlanken '-
	SetText	1506 - 1520	
	txtActOrdered_AfterUpdate	1529 - 1550	
	txtActOrdered_BeforeUpdate	1559 - 1580	
	txtActShipped_AfterUpdate	1589 - 1614	
	txtActShipped_BeforeUpdate	1623 - 1644	
	txtDate_AfterUpdate	1652 - 1670	
	txtDate_BeforeUpdate	1679 - 1709	
	txtQuantity_AfterUpdate	1719 - 1748	
frmSplash	(General Declarations)	1 - 20	' Module : Form_frmSplash ' Description: ' Procedures : cboLanguage_AfterUpdate() ' cmdCancel_Click() ' cmdContinue_Click() ' Form_Open(Cancel As Integer) ' Form_Timer() ' optUpgradeChoice_AfterUpdate() ' SetText()
	cboLanguage_AfterUpdate	21 - 42	' Comments : When the language is modified, update the system ' : variables, and the display. ' Parameters : ' Returns : ' Created : ' Modified : 01 Jul 1998 JSL ' ' --
	Form_Open	44 - 109	' Comments : On Open see if an upgrade is needed. If yes ' : give the user the option, if not act as a splash ' : screen. ' Parameters : ' Returns : ' Created : 06/29/98 JSL ' Modified : ' ' --
	Form_Timer	111 - 143	' Comments : if TimerInterval > 0 then count the iterations and ' : do do the correct events ' Parameters : ' Returns : ' Created : 06/29/98 JSL '

ModuleName	Procedure	Lines	Comments
			Modified : ' ' ---
	SetText	145 - 176	' Comments : Display the text in the proper language ' Parameters : ' Returns : ' Created : ' Modified : 01 Jul 1998 JSL ' ' ---
frmStock	(General Declarations)	1 - 42	'+ ' Module : Form_frmStock ' Description: Allows user to input inventory adjustments. When form ' opens, it finds the record in the list box that is closest ' to the current date. The user then has the option to edit ' delete, or add a record. ' Procedures : FindClosestRecord() ' cboSelect1_AfterUpdate() ' cboSelect1_GotFocus() ' cboSelect2_AfterUpdate() ' cboSelect2_GotFocus() ' cmdCount_Click() ' cmdDelete_Click() ' cmdNew_Click() ' cmdSave_Click() ' Form_Activate() ' Form_BeforeInsert(Cancel As Integer) ' Form_BeforeUpdate(Cancel As Integer) ' Form_Current() ' Form_Dirty(Cancel As Integer) ' Form_Error(DataErr As Integer, Response As Integer) ' Form_Open(Cancel As Integer) ' lstSelect_Click() ' SetText()
	cboSelect1_AfterUpdate	118 - 168	
	cboSelect1_GotFocus	176 - 195	
	cboSelect2_AfterUpdate	204 - 263	
	cboSelect2_GotFocus	273 - 292	
	cmdBack_Click	294 - 298	
	cmdCount_Click	307 - 369	
	cmdDelete_Click	379 - 431	
	cmdNew_Click	441 - 483	
	cmdSave_Click	492 - 547	
	FindClosestRecord	52 - 110	'+ ' Procedure : FindClosestRecord ' Comments : Find the record in the list box that is ' closest to the current date ' Parameters: - ' Created : 04 Feb 2000 Jeff Leiner ' Modified : 03 Jun 2003 LBlanken ' 16 Jun 2003 LBlanken '-
	Form_Activate	556 - 570	
	Form_BeforeInsert	579 - 604	
	Form_BeforeUpdate	613 - 662	
	Form_Current	671 - 699	

ModuleName	Procedure	Lines	Comments
	Form_Dirty	708 - 723	
	Form_Error	734 - 785	
	Form_Open	795 - 953	
	lstSelect_Click	962 - 996	
	SetText	1006 - 1020	
	txtAmount_GotFocus	1022 - 1025	
	txtDate_AfterUpdate	1031 - 1034	'mtracy 4/5/07 bugzilla 10924
	txtDate_GotFocus	1027 - 1029	
frmSuppliers	(General Declarations)	1 - 28	'+ ' Module : Form_frmSuppliers ' Description: Allows users to maintain the list of suppliers used ' throughout the system. ' Procedures : cmdDelete_Click() ' cmdNew_Click() ' cmdSave_Click() ' Form_Activate() ' Form_BeforeUpdate(Cancel As Integer) ' Form_Dirty(Cancel As Integer) ' Form_Error(DataErr As Integer, Response As Integer) ' Form_Open(Cancel As Integer) ' lstSelect_Click() ' SetText() ' txtName_AfterUpdate()
	cmdDelete_Click	37 - 93	'+ ' Procedure : cmdDelete_Click ' Comments : deletes the selected item in listbox ' Parameters: - ' Modified : 01 Jul 1998 JSL ' 03 Jun 2003 LBlanken ' 16 Jun 2003 LBlanken '-
	cmdNew_Click	103 - 135	
	cmdSave_Click	144 - 206	
	Form_Activate	215 - 230	
	Form_BeforeUpdate	239 - 280	
	Form_Dirty	289 - 304	
	Form_Error	314 - 352	
	Form_Open	363 - 429	
	lstSelect_Click	439 - 466	
	SetText	476 - 490	

ModuleName	Procedure	Lines	Comments
	txtName_AfterUpdate	499 - 516	
	UpdateControls	518 - 530	
frmTypes	(General Declarations)	1 - 4	
	Form_Open	5 - 13	
	lstMoveSelect1_afterupdate	37 - 39	
	SetText	21 - 35	
fsfrPipelineAction	(General Declarations)	1 - 19	'+ ' Module : Form_fsfrPipelineAction ' Description: ' Procedures : Form_Activate() ' Form_Open(Cancel As Integer) ' getEndYear() ' SetText() ' txtProduct_DblClick(Cancel As Integer) ' txtProduct_KeyDown(KeyCode As Integer, Shift As Integer)
	Form_Activate	27 - 96	'+ ' Procedure : Form_Activate ' Comments : move to correct record in table ' Parameters: - ' Modified : 01 Jul 1998 JSL ' 31 Jul 2003 LBlanken '-
	Form_Open	105 - 144	
	getEndYear	154 - 186	
	SetText	195 - 208	
	txtProduct_DblClick	217 - 246	
	txtProduct_KeyDown	255 - 270	
fsfrPipelineProblem	(General Declarations)	1 - 16	' Module : Form_fsfrPipelineProblem ' Description: ' Procedures : Form_Activate() ' Form_Open(Cancel As Integer) ' getEndYear() ' SetText() ' txtProduct_DblClick(Cancel As Integer)
	Form_Activate	17 - 81	' Comments : ' Parameters : ' Returns : ' Created : ' Modified : 01 Jul 1998 JSL ' ' ---
	Form_Open	83 - 120	' Comments : ' Parameters : Cancel ' Returns : ' Created : ' Modified : 01 Jul 1998 JSL ' ' ---
	getEndYear	122 - 156	' Comments : ' Parameters : ' Returns : ' Created : ' Modified : 01 Jul 1998 JSL ' ' ---
	SetText	158 - 177	' Comments : ' Parameters : ' Returns : ' Created : ' Modified : 01 Jul 1998 JSL ' ' ---
	txtProduct_DblClick	179 - 209	' Comments : ' Parameters : Cancel ' Returns : ' Created : ' Modified : 01 Jul 1998 JSL ' ' ---

ModuleName	Procedure	Lines	Comments
	txtProduct_KeyDown	211 - 215	
fsfrProcurementTable	(General Declarations)	1 - 16	'+ ' Module : Form_fsfrProcurementTable ' Description: ' Procedures : Form_Open(Cancel As Integer) ' Form_Current() ' setProductID() ' SetText() ' setYears()
	Form_Current	48 - 62	
	Form_Open	24 - 39	'+ ' Procedure : Form_Open ' Comments : set text, years and product ' Parameters: Cancel - ' Modified : 01 Jul 1998 JSL ' 28 Jul 2003 LBlanken '-
	setProductID	71 - 92	
	SetText	101 - 114	
	SetYears	123 - 143	
fsfrShipmentFunding	(General Declarations)	1 - 13	'+ ' Module : Form_fsfrShipmentSupplier ' Description: ' Procedures : Form_Open(Cancel As Integer) ' SetText()
	Form_Open	21 - 34	'+ ' Procedure : Form_Open ' Comments : set text on form ' Parameters: Cancel - ' Modified : 01 Jul 1998 JSL ' 31 Jul 2003 LBlanken '-
	SetText	43 - 56	
fsfrShipmentOrder	(General Declarations)	1 - 13	'+ ' Module : Form_fsfrShipmentOrder ' Description: ' Procedures : Form_Open(Cancel As Integer) ' SetText()
	Form_Open	21 - 34	'+ ' Procedure : Form_Open ' Comments : set text ' Parameters: Cancel - ' Modified : 01 Jul 1998 JSL ' 31 Jul 2003 LBlanken '-
	SetText	43 - 56	
fsfrShipmentProduct	(General Declarations)	1 - 13	'+ ' Module : Form_fsfrShipmentProduct ' Description: ' Procedures : Form_Open(Cancel As Integer) ' SetText()
	Form_Open	21 - 34	'+ ' Procedure : Form_Open ' Comments : set text ' Parameters: Cancel - ' Modified : 01 Jul 1998 JSL ' 31 Jul 2003 LBlanken '-
	SetText	43 - 56	
fsfrShipmentSupplier	(General Declarations)	1 - 13	'+ ' Module : Form_fsfrShipmentSupplier ' Description: ' Procedures : Form_Open(Cancel As Integer) ' SetText()
	Form_Open	21 - 34	'+ ' Procedure : Form_Open ' Comments : set text on form ' Parameters: Cancel - ' Modified : 01 Jul 1998 JSL ' 31 Jul 2003 LBlanken '-
	SetText	43 - 56	

ModuleName	Procedure	Lines	Comments
fsfrStatus_mon	(General Declarations)	1 - 28	'+ ' Module : Form_fsfrStatus_mon ' Description: ' Procedures : chkActFore_DblClick(Cancel As Integer) ' chkActFore_KeyDown(KeyCode As Integer, Shift As Integer) ' Form_Open(Cancel As Integer) ' SetText() ' txtActFore_DblClick(Cancel As Integer) ' txtBegBalance_DblClick(Cancel As Integer) ' txtBegBalance_KeyDown(KeyCode As Integer, Shift As Integer) ' txtConsumption_DblClick(Cancel As Integer) ' txtConsumption_KeyDown(KeyCode As Integer, Shift As Integer) ' txtQuantity_DblClick(Cancel As Integer) ' txtQuantity_KeyDown(KeyCode As Integer, Shift As Integer) ' txtStatus_DblClick(Cancel As Integer) ' txtStatus_KeyDown(KeyCode As Integer, Shift As Integer) ' txtStockAdj_DblClick(Cancel As Integer) ' txtStockAdj_KeyDown(KeyCode As Integer, Shift As Integer) ' txtSupplier_DblClick(Cancel As Integer) ' txtSupplier_KeyDown(KeyCode As Integer, Shift As Integer)
	chkActFore_DblClick	36 - 52	'+ ' Procedure : chkActFore_DblClick ' Comments : set globals and open consumption form ' Parameters: Cancel - ' Modified : 01 Jul 1998 JSL ' 12 Aug 2003 LBlanken '-
	chkActFore_KeyDown	61 - 76	
	Form_Open	85 - 98	
	SetText	107 - 120	
	txtActFore_DblClick	129 - 145	
	txtBegBalance_DblClick	154 - 169	
	txtBegBalance_KeyDown	178 - 193	
	txtConsumption_DblClick	202 - 219	
	txtConsumption_KeyDown	228 - 243	
	txtFunding_DblClick	245 - 270	
	txtFunding_KeyDown	273 - 288	
	txtQuantity_DblClick	298 - 322	
	txtQuantity_KeyDown	331 - 346	
	txtStatus_DblClick	355 - 379	
	txtStatus_KeyDown	388 - 403	

ModuleName	Procedure	Lines	Comments
	txtStockAdj_DblClick	412 - 428	
	txtStockAdj_KeyDown	437 - 452	
	txtSupplier_DblClick	461 - 485	
	txtSupplier_KeyDown	494 - 509	
fsfrStatus_qtr	(General Declarations)	1 - 28	'+ ' Module : Form_fsfrStatus_qtr ' Description: ' Procedures : chkActFore_DblClick(Cancel As Integer) ' chkActFore_KeyDown(KeyCode As Integer, Shift As Integer) ' Form_Open(Cancel As Integer) ' SetText() ' txtActFore_DblClick(Cancel As Integer) ' txtBegBalance_DblClick(Cancel As Integer) ' txtBegBalance_KeyDown(KeyCode As Integer, Shift As Integer) ' txtConsumption_DblClick(Cancel As Integer) ' txtConsumption_KeyDown(KeyCode As Integer, Shift As Integer) ' txtQuantity_DblClick(Cancel As Integer) ' txtQuantity_KeyDown(KeyCode As Integer, Shift As Integer) ' txtStatus_DblClick(Cancel As Integer) ' txtStatus_KeyDown(KeyCode As Integer, Shift As Integer) ' txtStockAdj_DblClick(Cancel As Integer) ' txtStockAdj_KeyDown(KeyCode As Integer, Shift As Integer) ' txtSupplier_DblClick(Cancel As Integer) ' txtSupplier_KeyDown(KeyCode As Integer, Shift As Integer)
	chkActFore_DblClick	36 - 52	'+ ' Procedure : chkActFore_DblClick ' Comments : set globals and open consumption form ' Parameters: Cancel - ' Modified : 01 Jul 1998 JSL ' 12 Aug 2003 LBlanken '-
	chkActFore_KeyDown	61 - 76	
	Form_Open	85 - 98	
	SetText	107 - 120	
	txtActFore_DblClick	129 - 145	
	txtBegBalance_DblClick	154 - 169	
	txtBegBalance_KeyDown	178 - 194	
	txtConsumption_DblClick	203 - 219	
	txtConsumption_KeyDown	228 - 243	
	txtQuantity_DblClick	252 - 277	
	txtQuantity_KeyDown	286 - 301	

ModuleName	Procedure	Lines	Comments
	txtStatus_DblClick	310 - 335	
	txtStatus_KeyDown	344 - 359	
	txtStockAdj_DblClick	368 - 384	
	txtStockAdj_KeyDown	393 - 408	
	txtSupplier_DblClick	417 - 442	
	txtSupplier_KeyDown	451 - 466	
fsfrStatusMethod	(General Declarations)	1 - 29	'+ ' Module : Form_fsfrStatus_mon ' Description: ' Procedures : chkActFore_DblClick(Cancel As Integer) ' chkActFore_KeyDown(KeyCode As Integer, Shift As Integer) ' Form_Open(Cancel As Integer) ' SetText() ' txtActFore_DblClick(Cancel As Integer) ' txtBegBalance_DblClick(Cancel As Integer) ' txtBegBalance_KeyDown(KeyCode As Integer, Shift As Integer) ' txtConsumption_DblClick(Cancel As Integer) ' txtConsumption_KeyDown(KeyCode As Integer, Shift As Integer) ' txtQuantity_DblClick(Cancel As Integer) ' txtQuantity_KeyDown(KeyCode As Integer, Shift As Integer) ' txtStatus_DblClick(Cancel As Integer) ' txtStatus_KeyDown(KeyCode As Integer, Shift As Integer) ' txtStockAdj_DblClick(Cancel As Integer) ' txtStockAdj_KeyDown(KeyCode As Integer, Shift As Integer) ' txtSupplier_DblClick(Cancel As Integer) ' txtSupplier_KeyDown(KeyCode As Integer, Shift As Integer)
	Form_Open	37 - 50	'+ ' Procedure : Form_Open ' Comments : set text on report ' Parameters: Cancel - ' Modified : 01 Jul 1998 JSL ' 12 Aug 2003 LBlanken '-
	SetText	59 - 72	
fsfrStatusMethod_mon	(General Declarations)	1 - 28	'+ ' Module : Form_fsfrStatusMethod_mon ' Description: ' Procedures : chkActFore_DblClick(Cancel As Integer) ' chkActFore_KeyDown(KeyCode As Integer, Shift As Integer) ' Form_Open(Cancel As Integer) ' SetText() ' txtActFore_DblClick(Cancel As Integer) ' txtBegBalance_DblClick(Cancel As Integer) ' txtBegBalance_KeyDown(KeyCode As Integer, Shift As Integer) ' txtConsumption_DblClick(Cancel As Integer) ' txtConsumption_KeyDown(KeyCode As Integer, Shift As Integer) ' txtQuantity_DblClick(Cancel As Integer) ' txtQuantity_KeyDown(KeyCode As Integer, Shift As Integer) ' txtStatus_DblClick(Cancel As Integer) ' txtStatus_KeyDown(KeyCode As Integer, Shift As Integer) '

ModuleName	Procedure	Lines	Comments
			txtStockAdj_DblClick(Cancel As Integer) ' txtStockAdj_KeyDown(KeyCode As Integer, Shift As Integer) ' txtSupplier_DblClick(Cancel As Integer) ' txtSupplier_KeyDown(KeyCode As Integer, Shift As Integer)
	chkActFore_DblClick	36 - 52	'+ ' Procedure : chkActFore_DblClick ' Comments : set globals and open consumption form ' Parameters: Cancel - ' Modified : 01 Jul 1998 JSL ' 12 Aug 2003 LBlanken '-
	chkActFore_KeyDown	61 - 76	
	Form_Open	85 - 98	
	SetText	107 - 120	
	txtActFore_DblClick	129 - 145	
	txtBegBalance_DblClick	154 - 169	
	txtBegBalance_KeyDown	178 - 193	
	txtConsumption_DblClick	202 - 218	
	txtConsumption_KeyDown	227 - 242	
	txtQuantity_DblClick	251 - 275	
	txtQuantity_KeyDown	284 - 299	
	txtStatus_DblClick	308 - 332	
	txtStatus_KeyDown	341 - 356	
	txtStockAdj_DblClick	365 - 381	
	txtStockAdj_KeyDown	390 - 405	
	txtSupplier_DblClick	414 - 438	
	txtSupplier_KeyDown	447 - 462	
fsfrStatusMethod_qtr	(General Declarations)	1 - 28	'+ ' Module : Form_fsfrStatusMethod_qtr ' Description: ' Procedures : chkActFore_DblClick(Cancel As Integer) ' chkActFore_KeyDown(KeyCode As Integer, Shift As Integer) ' Form_Open(Cancel As Integer) ' SetText() ' txtActFore_DblClick(Cancel As Integer) ' txtBegBalance_DblClick(Cancel As Integer) ' txtBegBalance_KeyDown(KeyCode As Integer, Shift As Integer) ' txtConsumption_DblClick(Cancel As Integer) ' txtConsumption_KeyDown(KeyCode As Integer, Shift As Integer) ' txtQuantity_DblClick(Cancel As Integer) ' txtQuantity_KeyDown(KeyCode

ModuleName	Procedure	Lines	Comments
			As Integer, Shift As Integer) ' txtStatus_DblClick(Cancel As Integer) ' txtStatus_KeyDown(KeyCode As Integer, Shift As Integer) ' txtStockAdj_DblClick(Cancel As Integer) ' txtStockAdj_KeyDown(KeyCode As Integer, Shift As Integer) ' txtSupplier_DblClick(Cancel As Integer) ' txtSupplier_KeyDown(KeyCode As Integer, Shift As Integer)
	chkActFore_DblClick	36 - 52	'+ ' Procedure : chkActFore_DblClick ' Comments : set globals and open consumption form ' Parameters: Cancel - ' Modified : 01 Jul 1998 JSL ' 12 Aug 2003 LBlanken '-
	chkActFore_KeyDown	61 - 76	
	Form_Open	85 - 98	
	SetText	107 - 120	
	txtActFore_DblClick	129 - 145	
	txtBegBalance_DblClick	154 - 169	
	txtBegBalance_KeyDown	178 - 193	
	txtConsumption_DblClick	202 - 218	
	txtConsumption_KeyDown	227 - 242	
	txtQuantity_DblClick	251 - 275	
	txtQuantity_KeyDown	284 - 299	
	txtStatus_DblClick	308 - 332	
	txtStatus_KeyDown	341 - 356	
	txtStockAdj_DblClick	365 - 381	
	txtStockAdj_KeyDown	390 - 405	
	txtSupplier_DblClick	414 - 438	
	txtSupplier_KeyDown	447 - 462	
fsfrStatusProduct	(General Declarations)	1 - 29	'+ ' Module : Form_fsfrStatus_mon ' Description: ' Procedures : chkActFore_DblClick(Cancel As Integer) ' chkActFore_KeyDown(KeyCode As Integer, Shift As Integer) ' Form_Open(Cancel As Integer) ' SetText() ' txtActFore_DblClick(Cancel As Integer) ' txtBegBalance_DblClick(Cancel As Integer) ' txtBegBalance_KeyDown(KeyCode As Integer, Shift As Integer) ' txtConsumption_DblClick(Cancel As Integer) '

ModuleName	Procedure	Lines	Comments
			txtConsumption_KeyDown(KeyCode As Integer, Shift As Integer) ' txtQuantity_DblClick(Cancel As Integer) ' txtQuantity_KeyDown(KeyCode As Integer, Shift As Integer) ' txtStatus_DblClick(Cancel As Integer) ' txtStatus_KeyDown(KeyCode As Integer, Shift As Integer) ' txtStockAdj_DblClick(Cancel As Integer) ' txtStockAdj_KeyDown(KeyCode As Integer, Shift As Integer) ' txtSupplier_DblClick(Cancel As Integer) ' txtSupplier_KeyDown(KeyCode As Integer, Shift As Integer)
	Form_Open	37 - 50	'+ ' Procedure : Form_Open ' Comments : set text on report ' Parameters: Cancel - ' Modified : 01 Jul 1998 JSL ' 12 Aug 2003 LBlanken '-
	SetText	59 - 72	

Reports

Table 18 - List of Reports

ModuleName	Procedure	Lines	Comments
rptAMCResult	(General Declarations)	1 - 1	
	Report_Close	53 - 86	' Comments : ' Parameters : ' Returns : ' Created : ' Modified : 01 Jul 1998 JSL ' ' -- ' : June 26, 2009 mahmed ' enable on Application close button on close. '-
	Report_Open	25 - 51	' Comments : ' Parameters : Cancel ' Returns : ' Created : ' Modified : 01 Jul 1998 JSL ' ' --
	SetText	9 - 23	'+ ' Procedure : SetText ' Comments : Set text on report ' Parameters: - ' Modified : 01 Jul 1998 JSL ' 11 Jul 2003 LBlanken '-
rptAnnualCosts	(General Declarations)	1 - 14	' Module : Report_rptAnnualCosts ' Description: ' Procedures : GroupHeader5_Print(Cancel As Integer, PrintCount As Integer) ' Report_Close() ' Report_Open(Cancel As Integer) ' SetText()
	Detail1_Format	15 - 22	
	GroupHeader5_Print	24 - 58	' Comments : ' Parameters : Cancel ' PrintCount ' Returns : ' Created : ' Modified : 01 Jul 1998 JSL ' ' --
	Report_Close	60 - 88	' Comments : ' Parameters : ' Returns : ' Created : ' Modified : 01 Jul 1998 JSL ' ' -- ' : June 26, 2009 mahmed ' enable on Application close button on close. '-

ModuleName	Procedure	Lines	Comments
	Report_Open	90 - 115	' Comments : ' Parameters : Cancel ' Returns : ' Created : ' Modified : 01 Jul 1998 JSL ' ' --
	SetText	117 - 187	' Comments : ' Parameters : ' Returns : ' Created : 06/16/98 JSL ' Modified : ' ' July 21, 2009 mahmed ' report subtitle now created from user-defined range ' --
rptAnnualCostsByFundingSource	(General Declarations)	1 - 14	' Module : Report_rptAnnualCosts ' Description: ' Procedures : GroupHeader5_Print(Cancel As Integer, PrintCount As Integer) ' Report_Close() ' Report_Open(Cancel As Integer) ' SetText()
	Detail1_Format	15 - 22	
	GroupHeader0_Print	24 - 62	' Comments : ' Parameters : Cancel ' PrintCount ' Returns : ' Created : ' Modified : 01 Jul 1998 JSL ' ' --
	GroupHeader5_Print	64 - 101	' Comments : ' Parameters : Cancel ' PrintCount ' Returns : ' Created : ' Modified : 01 Jul 1998 JSL ' ' --
	Report_Close	103 - 131	' Comments : ' Parameters : ' Returns : ' Created : ' Modified : 01 Jul 1998 JSL ' ' -- ' : June 26, 2009 mahmed ' enable on Application close button on close. '-
	Report_Open	133 - 158	' Comments : ' Parameters : Cancel ' Returns : ' Created : ' Modified : 01 Jul 1998 JSL ' ' --
	SetText	160 - 230	' Comments : ' Parameters : ' Returns : ' Created : 06/16/98 JSL ' Modified : ' ' July 21, 2009 mahmed ' report subtitle now created from user-defined range ' --
rptAnnualCostsBySupplier	(General Declarations)	1 - 14	' Module : Report_rptAnnualCosts ' Description: ' Procedures : GroupHeader5_Print(Cancel As Integer, PrintCount As Integer) ' Report_Close() ' Report_Open(Cancel As Integer) ' SetText()
	Detail1_Format	15 - 22	
	GroupHeader0_Print	24 - 62	' Comments : ' Parameters : Cancel ' PrintCount ' Returns : ' Created : ' Modified : 01 Jul 1998 JSL ' ' --
	GroupHeader5_Print	64 - 101	' Comments : ' Parameters : Cancel ' PrintCount ' Returns : ' Created : ' Modified : 01 Jul 1998 JSL ' ' --
	Report_Close	103 - 131	' Comments : ' Parameters : ' Returns : ' Created : ' Modified : 01 Jul 1998 JSL ' ' -- ' : June 26, 2009 mahmed ' enable on Application close button on close. '-
	Report_Open	133 -	' Comments : ' Parameters : Cancel ' Returns : ' Created : ' Modified : 01 Jul 1998

ModuleName	Procedure	Lines	Comments
		161	JSL ' ' ---
	SetText	163 - 233	' Comments : ' Parameters : ' Returns : ' Created : 06/16/98 JSL ' Modified : ' ' July 21, 2009 mahmed ' report subtitle now created from user-defined range ' --
rptCloneConsumptionFinal	(General Declarations)	1 - 4	
	Report_Close	81 - 119	' Comments : ' Parameters : ' Returns : ' Created : ' Modified : 01 Jul 1998 JSL ' ' --- ' : June 26, 2009 mahmed ' enable on Application close button on close. '-
	Report_Open	5 - 39	' Comments : ' Parameters : Cancel ' Returns : ' Created : ' Modified : 01 Jul 1998 JSL ' ' ---
	SetText	41 - 79	' Comments : ' Parameters : ' Returns : ' Created : 06/16/98 JSL ' Modified : ' ' July 21, 2009 mahmed ' report subtitle now created from user-defined range ' --
rptCloneConsumptionPreview	(General Declarations)	1 - 4	
	Report_Close	80 - 116	' Comments : ' Parameters : ' Returns : ' Created : ' Modified : 01 Jul 1998 JSL ' ' --- ' : June 26, 2009 mahmed ' enable on Application close button on close. '-
	Report_Open	5 - 38	' Comments : ' Parameters : Cancel ' Returns : ' Created : ' Modified : 01 Jul 1998 JSL ' ' ---
	SetText	40 - 78	' Comments : ' Parameters : ' Returns : ' Created : 06/16/98 JSL ' Modified : ' ' July 21, 2009 mahmed ' report subtitle now created from user-defined range ' --
rptGraphConsumption	(General Declarations)	1 - 31	'+ ' Module : Report_rptGraphConsumption ' Description: ' Procedures : PageHeader0_Format(Cancel As Integer, intFormatCount As Integer) ' Report_Close() ' Report_NoData(Cancel As Integer) ' Report_Open(Cancel As Integer) ' setRowSource() ' SetText()
	PageHeader0_Format	39 - 85	'+ ' Procedure : PageHeader0_Format ' Comments : ' Parameters: Cancel ' intFormatCount - ' Modified : 11 Jul 2003 LBlanken '-
	Report_Close	97 - 121	' Comments : ' Parameters: - ' Modified : ' ' ---
	Report_NoData	130 - 150	' Comments : ' Parameters: Cancel - ' Modified : ' ' ---
	Report_Open	159 - 217	' Comments : ' Parameters : cancel ' Returns : ' Created : ' Modified : 01 Jul 1998 JSL ' ' ---

ModuleName	Procedure	Lines	Comments
	Report_Page	282 - 284	
	setRowSource	226 - 252	' Comments : ' Parameters : ' Returns : ' Created : ' Modified : 01 Jul 1998 JSL ' ' --
	SetText	261 - 280	' Comments : ' Parameters : ' Returns : ' Created : ' Modified : 01 Jul 1998 JSL ' ' --
rptGraphCYP	(General Declarations)	1 - 16	' Module : Report_rptGraphCYP ' Description: ' Procedures : GetProductNames() ' PageHeader0_Format(Cancel As Integer, FormatCount As Integer) ' Report_Close() ' Report_Open(Cancel As Integer) ' setRowSource() ' SetText()
	GetProductNames	17 - 61	' Comments : ' Parameters: - ' Returns : String - ' Modified : ' ' --
	PageHeader0_Format	63 - 73	' Comments : ' Parameters: Cancel ' FormatCount - ' Modified : ' ' --
	Report_Close	75 - 94	' Comments : ' Parameters: - ' Modified : ' ' -- ' : June 26, 2009 mahmed ' enable on Application close button on close. '-
	Report_Open	96 - 155	' Comments : ' Parameters : cancel ' Returns : ' Created : ' Modified : 01 Jul 1998 JSL ' ' July 21, 2009 mahmed ' report subtitle now created from user-defined range ' --
	setRowSource	157 - 183	' Comments : ' Parameters : ' Returns : ' Created : ' Modified : 01 Jul 1998 JSL ' ' --
	SetText	185 - 206	' Comments : ' Parameters : ' Returns : ' Created : ' Modified : 01 Jul 1998 JSL ' ' --
rptGraphStockStatus	(General Declarations)	1 - 30	'+ ' Module : Report_rptGraphStockStatus ' Description: ' Procedures : FormHeader1_Format(Cancel As Integer, intFormatCount As Integer) ' PageHeader0_Format(Cancel As Integer, intFormatCount As Integer) ' Report_Close() ' Report_Open(Cancel As Integer) ' setRowSource() ' SetText()
	PageHeader0_Format	41 - 69	'+ '+ ' Procedure : PageHeader0_Format ' Comments : ' Parameters: Cancel ' intFormatCount - ' Modified : 11 Jul 2003 LBlanken ' July 21, 2009 mahmed ' report subtitle now created from user-defined range '-
	Report_Close	83 - 108	' Comments : ' Parameters: - ' Modified : ' ' --
	Report_Open	117 - 174	' Comments : ' Parameters : Cancel ' Returns : ' Created : ' Modified : 01 Jul 1998 JSL ' ' --
	setRowSource	183 -	' Comments : ' Parameters : ' Returns : ' Created : ' Modified : 01 Jul 1998 JSL ' ' ----

ModuleName	Procedure	Lines	Comments
		213	---
	SetText	222 - 241	' Comments : ' Parameters : ' Returns : ' Created : ' Modified : 01 Jul 1998 JSL ' ' ---
rptGraphTrend	(General Declarations)	1 - 16	' Module : Report_rptGraphTrend ' Description: ' Procedures : Detail1_Print(Cancel As Integer, PrintCount As Integer) ' GroupFooter5_Print(Cancel As Integer, PrintCount As Integer) ' GroupFooter6_Print(Cancel As Integer, PrintCount As Integer) ' Report_Close() ' Report_NoData(Cancel As Integer) ' Report_Open(Cancel As Integer) ' SetText()
	Detail1_Print	17 - 28	' Comments : ' Parameters: Cancel ' PrintCount - ' Modified : ' ' ---
	GroupFooter5_Print	30 - 41	' Comments : ' Parameters: Cancel ' PrintCount - ' Modified : ' ' ---
	GroupFooter6_Print	43 - 54	' Comments : ' Parameters: Cancel ' PrintCount - ' Modified : ' ' ---
	PageHeader0_Format	56 - 74	'mtracy 2/6/07 bugzilla 14399
	Report_Close	76 - 94	' Comments : ' Parameters: - ' Modified : ' ' --- ' : June 26, 2009 mahmed ' enable on Application close button on close. '-
	Report_NoData	96 - 105	' Comments : ' Parameters: Cancel - ' Modified : ' ' ---
	Report_Open	107 - 211	' Comments : ' Parameters: Cancel - ' Modified : ' ' ---
	SetText	213 - 223	' Comments : ' Parameters: - ' Modified : ' ' ---
rptGraphTrendCons	(General Declarations)	1 - 16	' Module : Report_rptGraphTrendCons ' Description: ' Procedures : Detail1_Print(Cancel As Integer, PrintCount As Integer) ' GroupFooter5_Print(Cancel As Integer, PrintCount As Integer) ' GroupFooter6_Print(Cancel As Integer, PrintCount As Integer) ' Report_Close() ' Report_NoData(Cancel As Integer) ' Report_Open(Cancel As Integer) ' SetText()
	Detail1_Print	17 - 28	' Comments : ' Parameters: Cancel ' PrintCount - ' Modified : ' ' ---
	GroupFooter5_Print	30 - 41	' Comments : ' Parameters: Cancel ' PrintCount - ' Modified : ' ' ---
	GroupFooter6_Print	43 - 54	' Comments : ' Parameters: Cancel ' PrintCount - ' Modified : ' ' ---------------------------

ModuleName	Procedure	Lines	Comments

	PageHeader0_Format	56 - 74	'mtracy 2/6/07 bugzilla 14399
	Report_Close	76 - 94	' Comments : ' Parameters: - ' Modified : ' ' -- ' : June 26, 2009 mahmed ' enable on Application close button on close. '-
	Report_NoData	96 - 105	' Comments : ' Parameters: Cancel - ' Modified : ' ' -- ------
	Report_Open	107 - 210	' Comments : ' Parameters: Cancel - ' Modified : ' ' -- ------
	SetText	212 - 222	' Comments : ' Parameters: - ' Modified : ' ' --
rptImport	(General Declarations)	1 - 26	'+ ' Module : Report_rptiMPORT ' Description: ' Procedures : GroupHeader1_Format(Cancel As Integer, FormatCount As Integer) ' Report_Close() ' Report_Open(Cancel As Integer) ' SetText()
	Report_Close	27 - 49	
	Report_Open	59 - 85	
	SetText	95 - 110	
rptInterpolate	(General Declarations)	1 - 14	' Module : Report_rptPipelineProblem ' Description: ' Procedures : Report_Close() ' Report_Open(Cancel As Integer) ' SetText()
	PageHeader0_Format	15 - 17	
	Report_Close	19 - 34	' Comments : ' Parameters: - ' Modified : ' ' --
	Report_Open	36 - 58	' Comments : ' Parameters : Cancel ' Returns : ' Created : ' Modified : 01 Jul 1998 JSL ' ' --
	SetText	60 - 79	' Comments : ' Parameters : ' Returns : ' Created : ' Modified : 01 Jul 1998 JSL ' ' ---- --
rptPipelineAction	(General Declarations)	1 - 14	' Module : Report_rptPipelineAction ' Description: ' Procedures : cmdDisplayGraph_Click() ' Report_Close() ' Report_Open(Cancel As Integer) ' SetText()
	Detail0_Format	15 - 21	
	Report_Close	23 - 42	' Comments : ' Parameters: - ' Modified : ' ' -- ' : June 26, 2009 mahmed ' enable on Application close button on close. '-
	Report_Open	44 - 69	' Comments : ' Parameters : Cancel ' Returns : ' Created : ' Modified : 01 Jul 1998 JSL ' ' --

ModuleName	Procedure	Lines	Comments
	SetText	71 - 97	' Comments : ' Parameters : ' Returns : ' Created : ' Modified : 01 Jul 1998 JSL ' ' --
rptPipelineProblem	(General Declarations)	1 - 13	' Module : Report_rptPipelineProblem ' Description: ' Procedures : Report_Close() ' Report_Open(Cancel As Integer) ' SetText()
	Report_Close	14 - 32	' Comments : ' Parameters: - ' Modified : ' ' -- ' : June 26, 2009 mahmed ' enable on Application close button on close. '-
	Report_Open	34 - 58	' Comments : ' Parameters : Cancel ' Returns : ' Created : ' Modified : 01 Jul 1998 JSL ' ' --
	SetText	60 - 79	' Comments : ' Parameters : ' Returns : ' Created : ' Modified : 01 Jul 1998 JSL ' ' --
rptProcurementTable	(General Declarations)	1 - 36	'+ ' Module : Report_rptProcurementTable ' Description: ' Procedures : Report_Close() ' Report_Open(Cancel As Integer) ' SetText() ' setYears()
	Report_Close	37 - 59	
	Report_Open	68 - 107	
	SetText	116 - 136	' Comments : ' Parameters : ' Returns : ' Created : ' Modified : 01 Jul 1998 JSL ' ' --
	SetYears	145 - 170	' Comments : ' Parameters : ' Returns : ' Created : ' Modified : 01 Jul 1998 JSL ' ' --
rptShipmentCostByFunding	(General Declarations)	1 - 20	'+ ' Module : Report_rptShipmentCostBySupplier ' Description: ' Procedures : Detail_Format(Cancel As Integer, FormatCount As Integer) ' PageFooter2_Format(Cancel As Integer, FormatCount As Integer) ' Report_Close() ' Report_Open(Cancel As Integer) ' SetText()
	Detail_Format	29 - 50	'+ ' Procedure : Detail1_Format ' Comments : set flags for freight and cost ' Parameters: Cancel ' FormatCount - ' Modified : 01 Jul 1998 JSL ' 31 Jul 2003 LBlanken '-
	PageFooter2_Format	61 - 74	
	Report_Close	84 - 103	
	Report_Open	113 - 144	
	SetText	154 - 169	
rptShipmentCostByFundingPortrait	(General Declarations)	1 - 20	'+ ' Module : Report_rptShipmentCostBySupplier ' Description: ' Procedures : Detail_Format(Cancel As Integer, FormatCount As Integer) '

ModuleName	Procedure	Lines	Comments
			PageFooter2_Format(Cancel As Integer, FormatCount As Integer) ' Report_Close() ' Report_Open(Cancel As Integer) ' SetText()
	Detail_Format	29 - 50	'+ ' Procedure : Detail1_Format ' Comments : set flags for freight and cost ' Parameters: Cancel ' FormatCount - ' Modified : 01 Jul 1998 JSL ' 31 Jul 2003 LBlanken '-
	PageFooter2_Format	61 - 74	
	Report_Close	84 - 102	
	Report_Open	112 - 144	
	SetText	154 - 169	
rptShipmentCostByProduct	(General Declarations)	1 - 19	'+ ' Module : Report_rptShipmentCostByProduct ' Description: ' Procedures : Detail1_Format(Cancel As Integer, FormatCount As Integer) ' Report_Close() ' Report_Open(Cancel As Integer) ' SetText()
	Detail1_Format	28 - 49	'+ ' Procedure : Detail1_Format ' Comments : set flags for freight and cost ' Parameters: Cancel ' FormatCount - ' Modified : 01 Jul 1998 JSL ' 31 Jul 2003 LBlanken '-
	PageFooter2_Format	60 - 73	
	Report_Close	86 - 104	
	Report_Open	114 - 149	
	SetText	159 - 174	
rptShipmentCostByProductPortrait	(General Declarations)	1 - 19	'+ ' Module : Report_rptShipmentCostByProduct ' Description: ' Procedures : Detail1_Format(Cancel As Integer, FormatCount As Integer) ' Report_Close() ' Report_Open(Cancel As Integer) ' SetText()
	Detail1_Format	28 - 49	'+ ' Procedure : Detail1_Format ' Comments : set flags for freight and cost ' Parameters: Cancel ' FormatCount - ' Modified : 01 Jul 1998 JSL ' 31 Jul 2003 LBlanken '-
	PageFooter2_Format	60 - 73	
	Report_Close	86 - 105	
	Report_Open	115 - 151	

ModuleName	Procedure	Lines	Comments
	SetText	161 - 176	
rptShipmentCostBySupplier	(General Declarations)	1 - 20	'+ ' Module : Report_rptShipmentCostBySupplier ' Description: ' Procedures : Detail_Format(Cancel As Integer, FormatCount As Integer) ' PageFooter2_Format(Cancel As Integer, FormatCount As Integer) ' Report_Close() ' Report_Open(Cancel As Integer) ' SetText()
	Detail_Format	29 - 50	'+ ' Procedure : Detail1_Format ' Comments : set flags for freight and cost ' Parameters: Cancel ' FormatCount - ' Modified : 01 Jul 1998 JSL ' 31 Jul 2003 LBlanken '-
	PageFooter2_Format	61 - 74	
	Report_Close	87 - 105	
	Report_Open	115 - 149	
	SetText	159 - 174	
rptShipmentCostBySupplierPortrait	(General Declarations)	1 - 20	'+ ' Module : Report_rptShipmentCostBySupplier ' Description: ' Procedures : Detail_Format(Cancel As Integer, FormatCount As Integer) ' PageFooter2_Format(Cancel As Integer, FormatCount As Integer) ' Report_Close() ' Report_Open(Cancel As Integer) ' SetText()
	Detail_Format	29 - 50	'+ ' Procedure : Detail1_Format ' Comments : set flags for freight and cost ' Parameters: Cancel ' FormatCount - ' Modified : 01 Jul 1998 JSL ' 31 Jul 2003 LBlanken '-
	PageFooter2_Format	61 - 74	
	Report_Close	87 - 108	
	Report_Open	118 - 152	
	SetText	162 - 177	
rptShipmentOrders	(General Declarations)	1 - 21	'+ ' Module : Report_rptShipmentOrders ' Description: ' Procedures : Detail_Format(Cancel As Integer, FormatCount As Integer) ' GroupHeader1_Format(Cancel As Integer, FormatCount As Integer) ' PageFooter2_Format(Cancel As Integer, FormatCount As Integer) ' Report_Close() ' Report_Open(Cancel As Integer) ' SetText()

ModuleName	Procedure	Lines	Comments
	Detail_Format	30 - 51	'+ ' Procedure : Detail_Format ' Comments : set flags for freight and cost ' Parameters: Cancel ' FormatCount - ' Modified : 01 Jul 1998 JSL ' 31 Jul 2003 LBlanken '-
	GroupHeader1_Format	61 - 98	
	PageFooter2_Format	109 - 122	
	Report_Close	135 - 153	
	Report_Open	163 - 201	
	SetText	211 - 226	
rptStatus_mon	(General Declarations)	1 - 22	'+ ' Module : Report_rptStatus_mon ' Description: ' Procedures : Detail0_Format(Cancel As Integer, FormatCount As Integer) ' getAdjustAmounts() ' getShipRecords() ' PageFooter1_Format(Cancel As Integer, FormatCount As Integer) ' PageHeader0_Format(Cancel As Integer, FormatCount As Integer) ' Report_Close() ' Report_Open(Cancel As Integer) ' SetText()
	Detail0_Format	31 - 44	'+ ' Procedure : Detail0_Format ' Comments : get adjustment amounts ' Parameters: Cancel ' FormatCount - ' Modified : 01 Jul 1998 JSL ' 11 Jul 2003 LBlanken '-
	getAdjustAmounts	54 - 91	
	getShipRecords	101 - 157	
	PageFooter1_Format	168 - 182	
	PageHeader0_Format	194 - 221	
	Report_Close	233 - 251	
	Report_Open	261 - 290	
	SetText	300 - 314	
rptStatus_Qtr	(General Declarations)	1 - 19	'+ ' Module : Report_rptStatus_Qtr ' Description: ' Procedures :

ModuleName	Procedure	Lines	Comments
			PageFooter1_Format(Cancel As Integer, FormatCount As Integer) ' PageHeader0_Format(Cancel As Integer, FormatCount As Integer) ' Report_Close() ' Report_Open(Cancel As Integer) ' SetText()
	PageFooter1_Format	28 - 42	'+ ' Procedure : PageFooter1_Format ' Comments : show footer based on flag ' Parameters: Cancel ' FormatCount - ' Modified : 01 Jul 1998 JSL ' 11 Jul 2003 LBlanken '-
	PageHeader0_Format	52 - 78	
	Report_Close	90 - 109	
	Report_Open	119 - 148	
	SetText	158 - 173	
rptStatusAllProducts_mon	(General Declarations)	1 - 22	'+ ' Module : Report_rptStatus_mon ' Description: ' Procedures : Detail0_Format(Cancel As Integer, FormatCount As Integer) ' getAdjustAmounts() ' getShipRecords() ' PageFooter1_Format(Cancel As Integer, FormatCount As Integer) ' PageHeader0_Format(Cancel As Integer, FormatCount As Integer) ' Report_Close() ' Report_Open(Cancel As Integer) ' SetText()
	Detail0_Format	31 - 44	'+ ' Procedure : Detail0_Format ' Comments : get adjustment amounts ' Parameters: Cancel ' FormatCount - ' Modified : 01 Jul 1998 JSL ' 11 Jul 2003 LBlanken '-
	getAdjustAmounts	54 - 91	
	getShipRecords	101 - 157	
	GroupHeader1_Format	160 - 162	
	PageFooter1_Format	172 - 186	
	PageHeader0_Format	196 - 228	
	Report_Close	240 - 258	
	Report_Open	268 - 297	
	SetText	307 -	

ModuleName	Procedure	Lines	Comments
		321	
rptStatusMatrix	(General Declarations)	1 - 14	' Module : Report_rptAnnualCosts ' Description: ' Procedures : GroupHeader5_Print(Cancel As Integer, PrintCount As Integer) ' Report_Close() ' Report_Open(Cancel As Integer) ' SetText()
	Detail1_Format	15 - 28	
	Detail1_Print	109 - 122	
	FormatCell	227 - 273	' Comments : ' Anytime the MOS calculation showed stock levels of a product above max, the number in that cell would appear in italics. If the MOS calculation showed stocks below min, the cell holding the MOS for that particular product would be highlighted. When the product was stocked out, the cell would be highlighted more strongly. ' Parameters : ' Returns : ' Created : ' ---
	GroupHeader0_Print	30 - 68	' Comments : ' Parameters : Cancel ' PrintCount ' Returns : ' Created : ' Modified : 01 Jul 1998 JSL ' ' ---
	GroupHeader5_Print	70 - 107	' Comments : ' Parameters : Cancel ' PrintCount ' Returns : ' Created : ' Modified : 01 Jul 1998 JSL ' ' ---
	PageHeader0_Format	124 - 131	
	Report_Close	133 - 162	' Comments : ' Parameters : ' Returns : ' Created : ' Modified : 01 Jul 1998 JSL ' ' --- ' : June 26, 2009 mahmed ' enable on Application close button on close. '-
	Report_Open	164 - 190	' Comments : ' Parameters : Cancel ' Returns : ' Created : ' Modified : 01 Jul 1998 JSL ' ' ---
	SetText	192 - 222	' Comments : ' Parameters : ' Returns : ' Created : 06/16/98 JSL ' Modified : ' ' July 21, 2009 mahmed ' report subtitle now created from user-defined range ' ---
rptStatusMatrixMethod	(General Declarations)	1 - 11	' Module : Report_rptStatusMatrixMethod ' Description: ' Procedures :
	Detail1_Format	12 - 42	
	FormatCell	248 - 288	' Comments : ' Anytime the MOS calculation showed stock levels of a product above max, the number in that cell would appear in italics. If the MOS calculation showed stocks below min, the cell holding the MOS for that particular product would be highlighted. When the product was stocked out, the cell would be highlighted more strongly. ' Parameters : ' Returns : ' Created : ' --

ModuleName	Procedure	Lines	Comments

	GroupHeader0_Print	44 - 82	' Comments : ' Parameters : Cancel ' PrintCount ' Returns : ' Created : ' Modified : 01 Jul 1998 JSL ' ' --
	GroupHeader5_Print	84 - 121	' Comments : ' Parameters : Cancel ' PrintCount ' Returns : ' Created : ' Modified : 01 Jul 1998 JSL ' ' --
	PageHeader0_Format	123 - 131	
	Report_Close	133 - 161	' Comments : ' Parameters : ' Returns : ' Created : ' Modified : 01 Jul 1998 JSL ' ' ---- -- ' : June 26, 2009 mahmed ' enable on Application close button on close. '-
	Report_NoData	163 - 183	' Comments : ' Parameters: Cancel - ' Modified : ' ' -- ------
	Report_Open	185 - 211	' Comments : ' Parameters : Cancel ' Returns : ' Created : ' Modified : 01 Jul 1998 JSL ' ' --
	SetText	213 - 243	' Comments : ' Parameters : ' Returns : ' Created : 06/16/98 JSL ' Modified : ' ' July 21, 2009 mahmed ' report subtitle now created from user-defined range ' -------------- --
rptStatusMatrixProduct	(General Declarations)	1 - 10	' Module : Report_rptStatusMatrixProduct ' Description: ' Procedures :
	Detail1_Format	11 - 41	
	FormatCell	251 - 294	' Comments : ' Anytime the MOS calculation showed stock levels of a product above max, ' the number in that cell would appear in italics. If the MOS calculation showed ' stocks below min, the cell holding the MOS for that particular product would be highlighted. ' When the product was stocked out, the cell would be highlighted more strongly. ' Parameters : ' Returns : ' Created : 16 Sep 2009 mahmed ' ------------------ ------------------------------------
	GroupHeader0_Print	43 - 81	' Comments : ' Parameters : Cancel ' PrintCount ' Returns : ' Created : ' Modified : 01 Jul 1998 JSL ' ' --
	GroupHeader5_Print	83 - 120	' Comments : ' Parameters : Cancel ' PrintCount ' Returns : ' Created : ' Modified : 01 Jul 1998 JSL ' ' --
	PageHeader0_Format	122 - 130	
	Report_Close	132 - 160	' Comments : ' Parameters : ' Returns : ' Created : ' Modified : 01 Jul 1998 JSL ' ' ---- -- ' : June 26, 2009 mahmed ' enable on

ModuleName	Procedure	Lines	Comments
			Application close button on close. '-
	Report_NoData	162 - 182	' Comments : ' Parameters: Cancel - ' Modified : ' ' ---
	Report_Open	186 - 213	' Comments : ' Parameters : Cancel ' Returns : ' Created : ' Modified : ' ' ---
	SetText	215 - 245	' Comments : ' Parameters : ' Returns : ' Created : 06/16/98 JSL ' Modified : ' ' July 21, 2009 mahmed ' report subtitle now created from user-defined range ' ---
rptStatusMethod_mon	(General Declarations)	1 - 22	'+ ' Module : Report_rptStatusMethod_mon ' Description: ' Procedures : Detail0_Format(Cancel As Integer, FormatCount As Integer) ' getAdjustAmounts() ' getShipRecords() ' PageFooter1_Format(Cancel As Integer, FormatCount As Integer) ' PageHeader0_Format(Cancel As Integer, FormatCount As Integer) ' Report_Close() ' Report_Open(Cancel As Integer) ' SetText()
	Detail0_Format	31 - 44	'+ ' Procedure : Detail0_Format ' Comments : get adjustment amounts ' Parameters: Cancel ' FormatCount - ' Modified : 01 Jul 1998 JSL ' 11 Jul 2003 LBlanken '-
	getAdjustAmounts	54 - 91	
	getShipRecords	101 - 157	
	PageFooter1_Format	168 - 182	
	PageHeader0_Format	194 - 226	
	Report_Close	238 - 258	
	Report_Open	268 - 297	
	SetText	307 - 321	
rptStatusMethod_QTR	(General Declarations)	1 - 22	'+ ' Module : Report_rptStatusMethod_qtr ' Description: ' Procedures : Detail0_Format(Cancel As Integer, FormatCount As Integer) ' getAdjustAmounts() ' getShipRecords() ' PageFooter1_Format(Cancel As Integer, FormatCount As Integer) ' PageHeader0_Format(Cancel As Integer, FormatCount As Integer) ' Report_Close() ' Report_Open(Cancel As Integer) ' SetText()

ModuleName	Procedure	Lines	Comments
	Detail0_Format	31 - 44	'+ ' Procedure : Detail0_Format ' Comments : get adjustment amounts ' Parameters: Cancel ' FormatCount - ' Modified : 01 Jul 1998 JSL ' 11 Jul 2003 LBlanken '-
	getAdjustAmounts	54 - 91	
	getShipRecords	101 - 157	
	PageFooter1_Format	168 - 182	
	PageHeader0_Format	194 - 227	
	Report_Close	238 - 256	
	Report_Open	266 - 295	
	SetText	305 - 319	
rptSupplyPlan	(General Declarations)	1 - 22	'+ ' Module : Report_rptStatus_mon ' Description: ' Procedures : Detail0_Format(Cancel As Integer, FormatCount As Integer) ' getAdjustAmounts() ' getShipRecords() ' PageFooter1_Format(Cancel As Integer, FormatCount As Integer) ' PageHeader0_Format(Cancel As Integer, FormatCount As Integer) ' Report_Close() ' Report_Open(Cancel As Integer) ' SetText()
	Detail0_Format	31 - 44	'+ ' Procedure : Detail0_Format ' Comments : get adjustment amounts ' Parameters: Cancel ' FormatCount - ' Modified : 01 Jul 1998 JSL ' 11 Jul 2003 LBlanken '-
	getAdjustAmounts	54 - 68	
	getShipRecords	78 - 134	
	PageFooter1_Format	145 - 158	
	PageHeader0_Format	170 - 196	
	Report_Close	205 - 223	
	Report_NoData	226 - 246	' Comments : ' Parameters: Cancel - ' Modified : ' ' --

ModuleName	Procedure	Lines	Comments
	Report_Open	255 - 284	
	SetText	294 - 308	
rptSupplyPlanBySupplier	(General Declarations)	1 - 22	'+ ' Module : Report_rptStatus_mon ' Description: ' Procedures : Detail0_Format(Cancel As Integer, FormatCount As Integer) ' getAdjustAmounts() ' getShipRecords() ' PageFooter1_Format(Cancel As Integer, FormatCount As Integer) ' PageHeader0_Format(Cancel As Integer, FormatCount As Integer) ' Report_Close() ' Report_Open(Cancel As Integer) ' SetText()
	Detail0_Format	31 - 44	'+ ' Procedure : Detail0_Format ' Comments : get adjustment amounts ' Parameters: Cancel ' FormatCount - ' Modified : 01 Jul 1998 JSL ' 11 Jul 2003 LBlanken '-
	getAdjustAmounts	54 - 68	
	getShipRecords	78 - 134	
	PageFooter1_Format	145 - 158	
	PageHeader0_Format	172 - 196	
	Report_Close	205 - 222	
	Report_Open	232 - 260	
	SetText	270 - 284	
rsubReportComments	(General Declarations)	1 - 12	' Module : Report_rsubReportComments ' Description: ' Procedures : SetText() ' Report_Open(Cancel As Integer)
	Report_Open	24 - 44	' Comments : ' Parameters: Cancel - ' Modified : ' ' --
	SetText	13 - 22	' Comments : ' Parameters: - ' Modified : ' ' --
rsubReportCommentsLS	(General Declarations)	1 - 12	' Module : Report_rsubReportComments ' Description: ' Procedures : SetText() ' Report_Open(Cancel As Integer)
	Report_Open	24 - 33	' Comments : ' Parameters: Cancel - ' Modified : ' ' --

ModuleName	Procedure	Lines	Comments
	SetText	13 - 22	' Comments : ' Parameters: - ' Modified : ' ' ---
srptImport	(General Declarations)	1 - 2	
	Report_Open	3 - 5	
	SetText	13 - 28	'+ ' Procedure : SetText ' Comments : set text on report ' Parameters: - ' Modified : 01 Jul 1998 JSL ' 31 Jul 2003 LBlanken '-
srptImportInfo	(General Declarations)	1 - 15	'+ ' Module : Report_rptiMPORT ' Description: ' Procedures : GroupHeader1_Format(Cancel As Integer, FormatCount As Integer) ' Report_Close() ' Report_Open(Cancel As Integer) ' SetText()
	GroupHeader1_Format	23 - 58	'+ ' Procedure : GroupHeader1_Format ' Comments : set captions for status' ' Parameters: Cancel ' FormatCount - ' Modified : 31 Jul 2003 LBlanken '-
	GroupHeader2_Format	60 - 64	
	Report_Open	72 - 92	'+ ' Procedure : Report_Open ' Comments : set captions and text of report and maximize ' Parameters: Cancel - ' Modified : 01 Jul 1998 JSL ' 31 Jul 2003 LBlanken '-
	SetText	102 - 117	
srptImportShipments	(General Declarations)	1 - 4	
	Report_Close	5 - 15	
	Report_Open	17 - 20	
	SetText	28 - 43	'+ ' Procedure : SetText ' Comments : set text on report ' Parameters: - ' Modified : 01 Jul 1998 JSL ' 31 Jul 2003 LBlanken '-
srptImportShipmentsImported	(General Declarations)	1 - 10	
	Detail_Format	11 - 32	
	Report_Open	34 - 36	
	ReportFooter_Format	62 - 64	
	SetText	44 - 60	'+ ' Procedure : SetText ' Comments : set text on report ' Parameters: - ' Modified : 01 Jul 1998 JSL ' 31 Jul 2003 LBlanken '-
srptImportShipmentsSkipped	(General Declarations)	1 - 10	
	Detail_Format	11 - 32	
	Report_Open	34 - 36	
	ReportFooter_Format	61 - 63	

ModuleName	Procedure	Lines	Comments
	SetText	44 - 59	'+ ' Procedure : SetText ' Comments : set text on report ' Parameters: - ' Modified : 01 Jul 1998 JSL ' 31 Jul 2003 LBlanken '-

Modules

Table 19 - List of Modules

ModuleName	Procedure	Lines	Comments
App Constants and Variables	(General Declarations)	1 - 72	'+ ' Module : App Constants and Variables ' Description: ' Procedures : ' Modified : ' 05/22/03 LBlanken Cleaned with Total Visual CodeTools ' 12 May 2004 LKB '-
basAccessConstants	(General Declarations)	1 - 220	'+ ' Module : basAccessConstants ' Description: ' Procedures : ' Modified : ' 05/22/03 LBlanken Cleaned with Total Visual CodeTools ' 12 May 2004 LKB '-
basAPIFunctions	(General Declarations)	1 - 253	'+ ' Module : basAPIFunctions ' Description: ' Procedures : ConvertTSB2Win(TSB_Struct As TSBAPI_OPENFILE, Win_Struct As TSBAPI_WINOPENFILENAME) ' ConvertWin2TSB(Win_Struct As TSBAPI_WINOPENFILENAME, TSB_Struct As TSBAPI_OPENFILE) ' getFileDialog_tsb(strInitialDir As String, strTitle As String, strFilter As String, strFile As String) ' GetOpenFile_TSB(strInitialDir As String, strTitle As String, strFilter As String, strFile As String) ' GetUserName_TSB() ' RemoveNulls_TSB(strIN As String) ' SZToString_TSB(strIN As String) ' Modified : ' 28 Jul 2003 LBlanken '-
	AppCloseEnabled	552 - 566	
	ConvertTSB2Win	261 - 302	'+ ' Procedure : ConvertTSB2Win ' Comments : Converts the passed TSBAPI structure to a Windows structure ' Parameters: TSB_Struct - record of type TSBAPI_OPENFILE ' Win_Struct - record of type TSBAPI_WINOPENFILENAME ' Modified : 28 Jul 2003 LBlanken '-
	ConvertWin2TSB	312 - 327	
	getFileDialog_tsb	342 - 381	
	GetOpenFile_TSB	394 - 441	
	GetUserName_TSB	451 - 478	
	RemoveNulls_TSB	488 - 510	

ModuleName	**Procedure**	**Lines**	**Comments**
	SZToString_TSB	521 - 542	
basAppConstants	(General Declarations)	1 - 131	'+ ' Module : App Constants ' Description: Constants used in the applications ' Procedures : ' Modified : 04 Sep 2003 LKB ' 1 Sep 2009 mahmed - add constants for shipment import function '-
basCheckAccessVersion	(General Declarations)	1 - 2	
	CheckAccessVersion	3 - 25	
basCloneConsumption	(General Declarations)	1 - 15	'+ ' Module : basCloneConsumption ' Description : ' Procedures : CheckCloneDataSource() ' CloneConsumption(strProductID As String)
	CheckCloneDataSource	22 - 48	'+ ' Procedure : CheckCloneDataSource ' Comments : Validates the Clone Data Source exists ' Parameters : - ' Modified : 18 Aug 2009 jsl '-
	CloneConsumption	58 - 99	
basCompactDB	(General Declarations)	1 - 19	'+ ' Module : basCompactDB ' Description: ' Procedures : CompactBackend() ' DoBackendCompact() ' GetBackendSize() ' GetBackendStartSize() ' GetDatabaseSize() ' SetBackendStartSize() ' SetCompactOnExit()
	CompactBackend	27 - 75	'+ ' Procedure : CompactBackend ' Comments : Determine if the Backend database has grown sufficiently to Compact Now. ' Parameters: - ' Created : 09-Jan-02 PM JSL ' Modified : 12 May 2004 LKB '-
	DoBackendCompact	85 - 98	
	GetBackendSize	108 - 130	
	GetBackendStartSize	140 - 154	
	GetDatabaseSize	164 - 184	
	SetBackendStartSize	194 - 214	
	SetCompactOnExit	225 - 255	
basCPTData	(General Declarations)	1 - 18	'+ ' Module : basCPTData ' Description: ' Procedures : GenCountryFile() ' GenCPTData() ' GenMethods() ' GenProducts() ' GenShipments(strYear As String) ' GenSupplierData()
	GenCountryFile	27 - 80	'+ ' Procedure : GenCountryFile ' Comments : To generate the Country File for the CPT Data. ' Parameters: - ' Returns : Integer - ' Created : 05/26/98 JSL ' Modified : 01 Jul 1998 JSL ' -
	GenCPTData	91 - 372	
	GenMethods	383 - 436	

ModuleName	Procedure	Lines	Comments
	GenProducts	447 - 504	
	GenShipments	516 - 604	
	GenSupplierData	614 - 666	
basExcelAdmin	(General Declarations)	1 - 70	
	ahtAddFilterItem	222 - 233	' Tack a new chunk onto the file filter. ' That is, take the old value, stick onto it the description, ' (like "Databases"), a null character, the skeleton ' (like "*.mdb;*.mda") and a final null character.
	ahtCommonFileOpenSave	125 - 220	' This is the entry point you'll use to call the common ' file open/save dialog. The parameters are listed ' below, and all are optional. ' ' In: ' Flags: one or more of the ahtOFN_* constants, OR'd together. ' InitialDir: the directory in which to first look ' Filter: a set of file filters, set up by calling ' AddFilterItem. See examples. ' FilterIndex: 1-based integer indicating which filter ' set to use, by default (1 if unspecified) ' DefaultExt: Extension to use if the user doesn't enter one. ' Only useful on file saves. ' FileName: Default value for the file name text box. ' DialogTitle: Title for the dialog. ' hWnd: parent window handle ' OpenFile: Boolean(True=Open File/False=Save As) ' Out: ' Return Value: Either Null or the selected filename
	GetOpenFile	88 - 123	' Here's an example that gets an Access database name.
	TestIt	71 - 86	' ' Dim Strfilter As String ' Dim lngFlags As Long ' Strfilter = ahtAddFilterItem(Strfilter, "Access Files (*.mda, *.mdb)", _ ' "*.MDA;*.MDB") ' Strfilter = ahtAddFilterItem(Strfilter, "dBASE Files (*.dbf)", "*.DBF") ' Strfilter = ahtAddFilterItem(Strfilter, "Text Files (*.txt)", "*.TXT") ' Strfilter = ahtAddFilterItem(Strfilter, "All Files (*.*)", "*.*") ' MsgBox "You selected: " & ahtCommonFileOpenSave(InitialDir:="C:\", _ ' filter:=Strfilter, FilterIndex:=3, Flags:=lngFlags, _ ' DialogTitle:="Hello! Open Me!") ' ' Since you passed in a variable for lngFlags, ' ' the function places the output flags value in the variable. ' Debug.Print Hex(lngFlags)
	TrimNull	235 - 243	
basExportReports	(General Declarations)	1 - 13	
	DetectExcel	20 - 43	'+ ' Procedure : DetectExcel ' Comments : ' Parameters : - ' Modified : 08 Dec 2010 JSL '-
	ExportReport	53 - 426	
	GetExcel	435 - 476	
	TableExists	486 - 495	

ModuleName	Procedure	Lines	Comments
basFileFunctions	(General Declarations)	1 - 19	'+ ' Module : basFileFunctions ' Description: ' Procedures : DirExists_TSB(strDir As String) ' FileExists_TSB(strDest As String) ' GetCodeDBNamePath_TSB() ' GetNamePart_TSB(strIN As String) ' GetPathPart_TSB(strPath As String) ' GetPmppPath() ' ParsePath_TSB(strIN As String, strDrive As String, strPath As String, strFileName As String, strExtension As String)
	DirExists_TSB	27 - 41	'+ ' Procedure : DirExists_TSB ' Comments : determines if the named directory exists ' Parameters: strDir - directory to check ' Returns : True - directory exists, False otherwise ' Modified : 12 May 2004 LKB ' -
	FileExists_TSB	52 - 61	
	GetCodeDBNamePath_TSB	71 - 89	
	GetNamePart_TSB	99 - 124	
	GetPathPart_TSB	134 - 156	
	GetPmppPath	167 - 192	
	ParsePath_TSB	206 - 264	
basFormsUtils	(General Declarations)	1 - 39	'+ ' Module : basFormsUtils ' Description: Utility function run from forms ' Procedures : ApplyFilter(frmIn As Form) ' BuildTmpSort(lngChild As Long ' CountryFormHead() As String ' DisableFrmCtls(frmIn As Form, intSection As Integer, fEnable As Boolean) ' FindCategoryChild(varParent As Variant) ' GenerateCodes(strName As String, strForm As String) ' getText(strTextID As String) ' IsLoaded(strFormName As String) ' IsLoadedRep(strRepName As String) ' listboxReset(lstBox As ListBox) ' MakeUSDATE(varDateIn As Variant) ' OpenCostReport(strByType As String) ' OpenDB(strDB As String) ' OpenPipeLineSummary() ' ProgFormHead() ' SaveForm(Frmin As Form) ' SetCboSelect2(strParent As String, ctl As Control) ' ShowHideSubform(formIn As Form, strSubForm As String) ' xfertoForm(strToForm As String, strFromForm As String, intKillMe ' As Integer)
	AddShipment	47 - 122	'+ ' Procedure : AddShipment ' Comments : Sets date to end of month and cycles through items in listbox ' : creating a shipment record for each ' Parameters: - ' Modified : 27 Jun 2003 LBlanken '-
	AdjustForCase	131 - 217	
	ApplyFilter	227 - 260	
	ApplyReportFilter	270 - 303	

ModuleName	Procedure	Lines	Comments
	AskSaveFormPopup	1764 - 1782	
	BuildFromYears	315 - 360	
	BuildThroughYears	373 - 415	
	BuildTmpSort	425 - 455	
	BuildYears	467 - 494	
	CheckInterpolateForm	2199 - 2236	
	checkMonthSelected	2501 - 2516	
	CheckValue	506 - 527	
	CountryFormHead	538 - 560	
	CreateTablePointer	1984 - 2012	' Comments : Create a pointer on a attached table to open ' : a recordset as a table type ' Parameters : strTable - the table to create the link on ' Returns : Recordset - ' Created : ' Modified : ' ' --
	DisableFrmCtls	571 - 607	
	DomainLookup_TSB	1608 - 1652	' Comments : Returns a field value in a specified set of records ' Parameters: strDatabase - path and name of database to look in or "" (blank string) for the current database ' strField - name of the field to return ' strDomain - name of the table or query to search in ' strCriteria - string expression specifying the WHERE clause of the query or blank for no constraints (all records) ' Returns : value of the specified field as a variant or NULL if no records matching strCriteria are found '
	ErrorWithData	617 - 632	
	FindCategoryChild	642 - 688	
	GenerateCodes	700 - 784	
	getAdjustAmounts	2410 - 2437	
	getAdjustAmountsMethod	2440 - 2467	
	getAMCFlag	2476 -	

ModuleName	Procedure	Lines	Comments
		2489	
	getDefaultPeriod	2273 - 2295	' Comments : ' Parameters: ' Returns : Integer - ' Created : 08/19/09 15:36 mahmed ' Modified : ' ' --
	getFundingSourceList	2367 - 2407	
	getFundingSourceName	2344 - 2355	
	GetProductName	2251 - 2272	
	getText	796 - 839	
	IsLoaded	850 - 875	
	IsLoadedRep	886 - 908	
	listboxReset	921 - 970	
	MakeUSDATE	982 - 1002	
	MakeYears	1012 - 1062	
	OpenCostReport	1072 - 1099	
	OpenDB	1109 - 1141	
	OpenPipeLineSummary	1152 - 1261	
	ProgFormHead	1272 - 1286	
	RequerySubform	1296 - 1376	
	SanitizeFileNameOrPath	1694 - 1762	' Comments : Return name with invalid characters replaced by "_" ' All characters greater than ASCII 31 to be used except for the following: "*/:<>?\\| ' The name may not be only dots ' Parameters : Candidate file name ' Created : Jan 18, 2007 - lblanken - Bug: 13663 ' ---

ModuleName	Procedure	Lines	Comments
	SaveForm	1387 - 1416	
	SetCboSelect2	1427 - 1456	
	SetDateRanges	1799 - 1979	
	SetDropDownWidth	2298 - 2335	'+ ' Procedure : SetDropDownWidth ' Comments : sets list width property of list box to longest entry in the list ' Parameters: - ' created : 26 July 2009 MAhmed '-
	SetInterpolateForm	2022 - 2185	
	ShowHideMain	1499 - 1515	'+ ' Procedure : ShowHideMain ' Comments : Shows/Hides main based on global mode ' Parameters: FormIn - current form ' strSubForm - source of subform ' Modified : 27 Jun 2003 LBlanken '-
	ShowHideSubform	1466 - 1491	
	UpdateCommandButtons	1654 - 1693	
	UpdateControls	1526 - 1557	
	xfertoForm	1571 - 1606	
basGetExcel	(General Declarations)	1 - 2	
	getexcelpath	3 - 62	
basGetStatus	(General Declarations)	1 - 17	'+ ' Module : basgetstatus ' Description: Set the form and report controls captions, tooltips, styles ' and statusbar text also gets the status text and report text ' Procedures : SetFormStatusText(frmIn As Form, Optional bytIndex As Byte, ' Optional bytType As Byte) ' SetReportCaptions(rptIn As Report, Optional bytIndex As Byte) ' GetStatusText(strForm As String, strCTL As String, bytIndex ' As Byte) ' Modified : 22 May 2003 LBlanken Cleaned with Total Visual CodeTools ' 16 Jun 2003 LBlanken '-
	GetStatusText	29 - 80	'+ ' Procedure : GetStatusText ' Comments : returns the text for the statusbar for the control as found ' in table ' Parameters: strForm - form ' strCTL - control on form ' bytIndex - ' Returns : String - ' Created : 02 Aug

ModuleName	Procedure	Lines	Comments
			2000 Jeff Leiner ' Modified : 16 Jun 2003 LBlanken ' -
	SetFormStatusText	94 - 214	
	SetReportCaptions	227 - 315	
basImport	(General Declarations)	1 - 9	
	ClearProcessed	843 - 860	
	ClearProcessedShipments	1057 - 1074	
	ClearUnMapped	556 - 573	'+ ' Procedure : ClearUnMapped ' Comments : Removes all unmapped data from the Import tables ' Parameters: - ' Modified : 17 Jul 2006 MAT '-
	ImportForecastData	575 - 730	
	ImportProducts	10 - 549	
	ImportShipments	910 - 1049	
	Validate_Period	743 - 835	'+ ' Procedure : Validate_Period ' Comments : Returns true if at least part of time period entered is unique ' (accept edits). Returns false if at least part of time period ' not unique (reject edits). Unique means time period not duplicated ' by another record of the same type. Actual consumption can duplicate ' a forecast time period ' Parameters: - ' Returns : - ' Modified : 01 Jul 1998 JSL ' 16 Jun 2003 LBlanken ' -
	ValidateFile	862 - 908	
basImportNewvernshipments	(General Declarations)	1 - 14	'+ ' Module : basImportNewvernshipments ' Description: ' Procedures : GetNewvernShipmentFile()
	GetNewvernShipmentFile	23 - 161	'+ ' Procedure : GetNewvernShipmentFile ' Comments : Find a newvern shipment file and import it into the .DB ' Parameters: - ' Returns : Boolean - ' Created : 06-Mar-00 Jeff Leiner ' Modified : 12 May 2004 LKB ' -
basMiscTools	(General Declarations)	1 - 24	'+ ' Module : Misc_Tools ' Description: ' Procedures : BuildInString(lst As ListBox, fIsString As Boolean) ' CountOccurrences_TSB(strIn As String, strFind As String) ' FillString_TSB(strChar As String, intCount As Integer) ' MaxOfThree_TSB(varValue1 As Variant, varValue2 As Variant, varValue3 As Variant) ' Movepointer(frmActive As Form, varDirection As Variant) ' NullToValue(varNull As Variant, strValue As String) ' Open_Admin_Form(strFormName As String) ' Open_Form(strFormName As String) ' Open_Supervisor_Form(strFormName As String) ' Return_To_Form(strFormName As String) ' StripChars_TSB(strIn As String,

ModuleName	Procedure	Lines	Comments
			strStrip As String)
	BuildExclSupplierNameString	713 - 767	
	BuildInString	35 - 72	'+ ' Procedure : BuildInString ' Comments : Build a Instring statement from the selected values ' : in a multiselect listbox ' Parameters: lst - The listbox to use ' fIsString - If the value is a string it must be in single quote ' Returns : String - ' Created : 05 Apr 2000 Jeff Leiner ' Modified : 20 Jun 2003 LBlanken ' -
	ceiling	658 - 673	
	CountDelimitedWords_TSB	591 - 610	' Comments : returns the number of words in a delimited string ' Parameters: strIn - string to count words in ' chrDelimit - character that delimits words in strIn ' Returns : number of occurrences '
	CountOccurrences_TSB	83 - 122	
	FillString_TSB	133 - 156	
	GetDelimitedWord_TSB	613 - 656	' Comments : returns word intIndex in delimited string strIn ' Parameters : strIn - string to search ' intIndex - word position to find (value of 0 or greater than word count is undefined.) ' chrDelimit - character used as the delimter ' Returns : nth word '
	GetValue	882 - 899	
	IsCharLetter_TSB	512 - 531	' Comments : determines if the passed character is an letter (a-z, A-Z, à-ÿ, À-Þ) ' Parameters: varChar - Character as variant ' Returns : True - character is alpha, False - character is not alpha '
	IsCharNumeric_TSB	534 - 545	' Comments : determines if the passed character is numeric ' Parameters: varChar - character to check ' Out : True - character is numeric, False - character is not numeric '
	MakeDataSetMatrix	776 - 845	
	MaxOfThree_TSB	168 - 220	
	Movepointer	232 - 314	
	ntz	846 - 871	' Comments : Returns a 0 if the value is null or the same value if not null ' Useful in queries and calculations where there is missing data ' Parameters : varValue - value to test ' Returns : 0 if null, original value if not null ' Created : ' Modified : ' ' --
	NullToValue	326 - 340	
	Open_Admin_Form	351 - 367	

ModuleName	Procedure	Lines	Comments
	Open_Form	378 - 390	
	Open_Supervisor_Form	401 - 419	
	ReplaceString_TSB	484 - 508	' Comments : replaces a substring in a string with another ' Parameters : strTextIn - string to work on ' strFind - string to find ' strReplace - string to replace with ' fCaseSensitive - True for case sensitive search, False for case-insensitive search ' Returns : modified string '
	Return_To_Form	430 - 442	
	RoundtoNextNumber	674 - 701	' Comments : Rounds a number to a specified number of decimal places (if any ' : remainder than the value is rounded up). ' Parameters: dblValue - number to round ' intDecimals - number of decimal places to round to ' (positive for right of decimal, negative for left) ' Returns : Rounded number ' Created : ' Modified : ' ' --
	StringToArray_TSB	570 - 589	' Comments : Converts a delimited string to an array of words ' Parameters: strIn - string to convert ' arrIn - array of strings (1-based) ' chrDelimit - character used to delimit words in strIn ' Returns : number of words '
	StripChars_Special	546 - 566	
	StripChars_TSB	453 - 482	
basMsgBoxUnicode	(General Declarations)	1 - 16	'+ ' Module : basMsgBoxUnicode ' Description : Functions for producing a unicode compliant message box ' Procedures : MsgBoxU() ' Modified : 09 Nov 2005 JSL '-
	MsgBoxU	31 - 73	'+ ' Procedure : MsgBoxU ' Comments : Unicode enabled Messagebox wrapper, accepting all arguments ' : although help doesn't work at this time. ' Parameters : Prompt - the Message to display ' : lngButtons - the msgbox options (of vbMsgboxStyle) ' : Title - The title of the msgbox. if null then application name ' : - is displayed. ' : HelpFile = the path to the help file (N/A) ' : Context - the help context ID (N/A) ' Returns : Long - the result of the message box ' Created : 09 Nov 2005 jleiner ' Modified : '-
basPDFOutput	(General Declarations)	1 - 5	
	ChoosePDFPrinter	83 - 87	
	GetPDFEngine	122 - 146	
	IsReportLoaded	103 - 105	
	printreportPDF	6 - 79	
	SetDefaultPrinter	89 - 101	

ModuleName	Procedure	Lines	Comments
	SetOpen	107 - 111	
	SetPDFEngine	147 - 174	
basPriceFunctions	(General Declarations)	1 - 22	'+ ' Module : basPriceFunctions ' Description: ' Procedures : EstOrderDate(lngShipID As Long) ' GetDefaultSupplier(strProductID) ' getFreightRate(lngShipID As Long) ' getheader(Col) ' GetUnitvalue(lngShipment As Long) ' Round(ByVal dblNumber As Double, ByVal intDecimals As Integer) ' xtabprint(lngView As Long)
	EstOrderDate	32 - 71	'+ ' Procedure : EstOrderDate ' Comments : When a shipments Order Date is not specified caculate its ' its estimated Date and return the value ' Parameters: lngShipID - ' Returns : Variant - The Estimated Date ' Created : 06/10/98 JSL ' Modified : 12 May 2004 LKB ' -
	GetDefaultSupplier	82 - 104	
	getFreightRate	118 - 185	
	getheader	196 - 212	
	GetUnitvalue	222 - 294	
	Round	305 - 323	
	xtabprint	335 - 405	
	xtabprintByFunding	416 - 488	
	xtabprintBySupplier	498 - 570	
	xtabprintStatusMethod	657 - 725	
	xtabprintStatusProduct	579 - 646	'+ ' Procedure : xtabprintStatusProduct ' Comments : to Dynamically create the data for a report based on a ' : Crosstab query ' Parameters: lngView - ' Returns : - ' Created : 11/08/2009 mahmed ' -
basReferences	(General Declarations)	1 - 14	'+ ' Module : basReferences ' Description: ' Procedures : ReferenceInfo()
	ReferenceInfo	22 - 50	'+ ' Procedure : ReferenceInfo ' Comments : ' Parameters: - ' Returns : - ' Modified : 12 May 2004 LKB ' -
basRegistryExcel	(General Declarations)	1 - 90	
	fReturnRegKeyValue	91 - 167	
basRegistrySettings	(General Declarations)	1 - 111	'+ ' Module : basRegistrySettings ' Description: ' Procedures : GetAllowCPTExports() ' GetNewInstall() ' RegGetKeyValue_TSB(lngRootKey As Long, strKeyName As String, strValueName As String) ' RegSetKeyValue_TSB(lngRootKey As Long,

ModuleName	Procedure	Lines	Comments
			strKeyName As String, strValueName As String, varValue As Variant, lngValueType As Long) ' SetNewInstall(strValue As String)
	GetAllowCPTExports	119 - 141	'+ ' Procedure : GetAllowCPTExports ' Comments : ' Parameters: - ' Returns : Boolean - ' Modified : 12 May 2004 LKB ' -
	GetNewInstall	151 - 173	
	RegCreateNewKey_TSB	339 - 361	'+ ' Procedure : RegCreateNewKey_TSB ' Comments : Creates a new key ' Parameters: lngRootKey - root key value, must be one of the following constants ' regHKeyClassesRoot ' regHKeyCurrentUser ' regHKeyLocalMachine ' regHKeyUsers ' strKeyName - The name of the key to create ' Returns : True if successful, False otherwise ' Modified : 30 Jul 2003 JLeiner ' -
	RegDeleteKey_TSB	374 - 390	'+ ' Procedure : RegDeleteKey_TSB ' Comments : Deletes the specified key from the system registry ' Parameters: lngRootKey - root key value, must be one of the following constants ' regHKeyClassesRoot ' regHKeyCurrentUser ' regHKeyLocalMachine ' regHKeyUsers ' strKeyName - The name of the key to delete ' Returns : True if successful, False otherwise ' Modified : 30 Jul 2003 JLeiner ' -
	RegDeleteValue_TSB	406 - 428	
	RegGetKeyValue_TSB	189 - 251	
	RegSetKeyValue_TSB	269 - 301	'
	SetNewInstall	310 - 326	
basReportComments	(General Declarations)	1 - 16	'+ ' Module : basReportComments ' Description: ' Procedures : GetReportComments(intReportType As Integer, dtmStart As Date, dtmEnd As Date)
	GetReportComments	27 - 146	'+ ' Procedure : GetReportComments ' Comments : Producte a list (table) of all Printable comments that ' : fit the product /methods selected and the date range. ' Parameters: intReportType - 1 Consumption only, 2 Stock Status ' dtmStart ' dtmEnd - ' Created : 16-Feb-00 Jeff Leiner ' Modified : 26-Jan-04 LBlanken fix comment for stock '-
basResize	(General Declarations)	1 - 11	
	ResizeReportControls	12 - 62	
basSafeWrappers	(General Declarations)	1 - 13	
	SafeClng	111 - 123	

ModuleName	Procedure	Lines	Comments
	SafeCStr	42 - 54	
	SafeCVDate	89 - 101	
	SafeFormat	14 - 30	
	SafeNulltoString	134 - 146	
	SafeNZ	158 - 170	
	SafeTrim	183 - 201	
	SafeUCase	66 - 78	
basSaveFile	(General Declarations)	1 - 82	'+ ' Module : basSaveFile ' Description: ' Procedures : ConvertTSB2Win(TSB_Struct As TSBAPI_OPENFILE, Win_Struct As TSBAPI_WINOPENFILENAME) ' ConvertWin2TSB(Win_Struct As TSBAPI_WINOPENFILENAME, TSB_Struct As TSBAPI_OPENFILE) ' CreateFilterString_TSB(ParamArray varFilt() As Variant) ' GetSaveFile_TSB(strInitialDir As String, strTitle As String, strDefName As String) ' RemoveNulls_TSB(strIN As String)
	ConvertTSB2Win	90 - 131	'+ ' Procedure : ConvertTSB2Win ' Comments : Converts the passed TSBAPI structure to a Windows structure ' Parameters: TSB_Struct - record of type TSBAPI_OPENFILE ' Win_Struct - record of type TSBAPI_WINOPENFILENAME ' Modified : 12 May 2004 LKB '-
	ConvertWin2TSB	141 - 156	
	CreateFilterString_TSB	169 - 203	
	GetSaveFile_TSB	215 - 264	
	GetSaveFileXML_TSB	275 - 324	
	RemoveNulls_TSB	332 - 354	'+ ' Procedure : RemoveNulls_TSB ' Comments : Removes terminator from a string ' Parameters: strIN - string to modify ' Returns : String - modified string ' Modified : 12 May 2004 LKB ' -
basShortCuts	(General Declarations)	1 - 45	'+ ' Module : basShortCuts ' Description: ' Procedures : CopyAppShortcut() ' GetSpecialFolder(CSIDL As Long)
	CopyAppShortcut	53 - 98	'+ ' Procedure : CopyAppShortcut ' Comments : ' Parameters: - ' Returns : - ' Modified : 12 May 2004 LKB ' -
	GetSpecialFolder	108 - 141	
basStacks	(General Declarations)	1 - 23	'+ ' Module : basStacks ' Description: Contains functions for the creation of the stack used ' in tracking the progression of open forms in the database. '

ModuleName	Procedure	Lines	Comments
			Procedures : StackForms(strForm As String) ' UnstackForms() ' Created : 02 Apr 1999 Admin ' Modified : 23 Jul 1999 Admin Cleaned with CodeTools ' 30 May 2003 LBlanken as per code review ' 16 Jun 2003 LBlanken '-
	JumpForms	150 - 182	
	ResetSubform	33 - 89	'+ ' Procedure : ResetSubForm ' Comments : Used to open form as sfrSform on frmMain ' Used when double clicking item in report data ' Parameters: strForm - Form to set sfrSform to ' Returns : - ' Modified : 30 May 2003 LBlanken as per code review ' 16 Jun 2003 LBlanken ' -
	StackForms	101 - 141	
	UnstackForms	193 - 219	
basStyle	(General Declarations)	1 - 14	'+ ' Module : basStyle ' Description: ' Procedures : SetStyle(lngStyle As Long, ctlTemp As Control)
	SetStyle	25 - 106	'+ ' Procedure : SetStyle ' Comments : Sets the style based on table for forms and reports ' Parameters: lngStyle - unique identifier for style ' ctlTemp - control to apply style to ' Returns : - ' Modified : 30 May 2003 LBlanken as per code review ' 16 Jun 2003 LBlanken ' 18 AUG 2009 Jleiner - Added New Hightlighted Types ' -
basToolbarFunctions	(General Declarations)	1 - 49	'+ ' Module : basToolbarFunctions ' Description: ' Procedures : AddNewMB() ' CPTExport() ' DirtyContinue() ' FileClose() ' FileCopy_PL() ' FileMenu(p_strSelect As String) ' FileNew() ' FileOpen() ' ImportShipmentData() ' IsImportVisible() ' MakeCurrent() ' SampleMenuDisable() ' SetFormMode() ' SetProgramText() ' ToggleClickMe() ' UpdateBackEnd() ' MakeActive(p_fActive As Boolean, p_strCode As String)
	AddNewMB	59 - 142	'+ ' Procedure : AddNewMB ' Comments : ' Parameters: - ' Modified : 14 Oct 2003 LKB '- '** ' This procedure creates a new menu bar. '**
	ChangeProgram	153 - 196	
	CPTExport	205 - 222	
	DirtyContinue	232 - 260	
	FileClose	270 - 343	
	FileCopy_PL	353 - 439	

ModuleName	Procedure	Lines	Comments
	FileMenu	449 - 495	
	FileNew	505 - 531	'check if form is dirty and prompt to save (false only if user says cancel)
	FileOpen	542 - 866	
	HelpClose_TSB	895 - 915	' Comments : marks the specified help file as closed ' Parameters: strFile - name of help (*.HLP) file to mark ' Returns : True if successful, False otherwise '
	HelpContext_TSB	877 - 894	
	ImportShipmentData	924 - 945	
	IsImportVisible	957 - 988	
	LangMenus	999 - 1306	
	lblID	1316 - 1330	
	MakeActive	2039 - 2080	
	MakeCurrent	1340 - 1425	
	OpenAbout	1435 - 1447	
	OpenHelp	1457 - 1469	
	OpenLanguage	1479 - 1496	
	OpenProperties	1506 - 1518	
	RelinkToDefault	1528 - 1562	
	RunLanguageForm	1573 - 1627	
	SampleMenuDisable	1642 - 1655	
	SaveSubForm	1665 -	

ModuleName	Procedure	Lines	Comments
		1705	
	SetFormMode	1716 - 1743	
	SetProgramText	1753 - 1793	
	ToggleClickMe	1808 - 1828	
	UpdateAndMove	1838 - 1878	
	UpdateBackEnd	1889 - 1927	
	UpdateCommandBars	1937 - 2003	
	UpdateTreeviewLanguage	2013 - 2028	
basTranslation	(General Declarations)	1 - 20	'+ ' Module : basTranslation ' Description: ' Procedures : BuildTranslationTable() ' CaptureTabPageCaption() ' CreateCaptiontext() ' CreateFormTitleCaptions() ' CreateReportCaptiontext() ' CreateStatusbartext() ' CreateToolTipText() ' setButtonColor()
	1	92 - 111	
	1	149 - 167	
	1	216 - 232	
	1	273 - 291	
	2	387 - 406	
	3	443 - 456	
	4	329 - 348	
	BuildTranslationTable	28 - 55	'+ ' Procedure : BuildTranslationTable ' Comments : build the translation table based on forms ' Parameters: - ' Returns : - ' Modified : 12 May 2004 LKB ' -
	setButtonColor	465 - 502	
basTreeviewUtils	(General Declarations)	1 - 15	'+ ' Module : basTreeviewUtils ' Description: ' Procedures : CreateTreeview() ' SetActiveNode(strKey As String)

ModuleName	Procedure	Lines	Comments
	CreateParenttoRoot	26 - 106	'+ ' Procedure : CreateParenttoRoot ' Comments : For a given nodeID in the treeview, make sure it parent ' : nodes exist all to the root (ParentID = 0). This procedure ' : makes Recursive calls to itself. ' Parameters: lngNodeID - The parent node to test for ' : ctlTreeview - The Treeview being populated ' Returns : Boolean - True returns that the node exists. ' Modified : ' -
	CreateTreeview	116 - 174	
	SetActiveNode	183 - 201	
basUndoMainSubform	(General Declarations)	1 - 20	'+ ' Module : basUndoMainSubform ' Description: ' Procedures : SaveMain(Frmin As Form, strTempTable As String) ' SaveSub(Frmin As Form, strTempTable As String) ' UndoMain(Frmin As Form, strTempTable As String) ' UndoSub(Frmin As Form, strTempTable As String)
	SaveMain	29 - 83	'+ ' Procedure : SaveMain ' Comments : Save values in main form to temporary table ' Parameters: Frmin - the name of the form ' strTempTable - the name of the temporary table ' Modified : 30 May 2003 LBlanken as per code review ' 16 Jun 2003 LBlanken '-
	SaveSub	94 - 155	
	UndoMain	166 - 217	
	UndoSub	229 - 325	
basUpgradeBackEnd	(General Declarations)	1 - 18	'+ ' Module : basUpgradeBackEnd ' Description: ' Procedures : checkBackEndVersion(strDataDir As String) ' CopyIndexes_TSB(strDatabase As String, strDestination As String, strTable As String) ' CopyRelations_TSB(strDatabase As String, strDestination As String) ' CreateAttachement_TSB(strSourceDatabase As String, strDestDatabase As String, strTable As String) ' DeleteRelations_TSB(strDestination As String) ' UpgradeProgram(sProgram As String, sDir As String)
	checkBackEndVersion	31 - 107	'+ ' Procedure : checkBackEndVersion ' Comments : Compares the version number of the PMP_MPTY against ' : that of the globalmoh.MDB being checked. If the ' : files is found and the version numbers match then ' : the version is OK, else the files will need to be ' : updated. ' Parameters: strDataDir - ' Returns : Integer - ' Created : 06/04/98 JSL ' Modified : 12 May 2004 LKB ' -
	CopyIndexes_TSB	120 - 296	
	CopyRelations_TSB	307 - 400	

ModuleName	Procedure	Lines	Comments
	CreateAttachement_TSB	412 - 448	
	DeleteRelations_TSB	458 - 517	
	UpgradeProgram	529 - 755	
basUpgradeMDB_Utils	(General Declarations)	1 - 25	'+ ' Module : basUpgradeMDB_Utils ' Description: ' Procedures : ClearOldObjects(strUserFile As String) ' CreateUpgradeMDB(sPath As String) ' UpdateProgramTable()
	ClearOldObjects	26 - 123	
	CreateUpgradeMDB	134 - 192	
	UpdateProgramTable	204 - 289	
basWindowsVersion	(General Declarations)	1 - 20	
	fOSNameXP	21 - 36	
basXMLConsumptionImport	(General Declarations)	1 - 15	'+ ' Module : basXMLConsumptionImport ' Description: ' Procedures : ConsDataExists() ' ConsImport() ' OpenImport() ' OpenReconciliation()
	ConsDataExists	23 - 39	'+ ' Procedure : ConsDataExists ' Comments : checks if data exists in import tables ' Parameters: - ' Returns : Boolean - ' Modified : 21 Aug 2003 LKB ' -
	ConsImport	49 - 82	
	OpenImport	92 - 115	
	OpenReconciliation	125 - 142	
basXMLDump	(General Declarations)	1 - 8	
	CreateXMLFromTables	15 - 52	'+ ' Procedure : CreateXMLFromTables ' Comments : Create the Raw XML Dump based on the file ' Parameters : - ' Modified : 22 Mar 2007 JSL '-
	ExportProgramDataXML	61 - 115	
	MemotoString	125 - 138	
	TransformXML	147 - 175	
basXMLForecastImport	(General Declarations)	1 - 15	'+ ' Module : basXMLForecastImport ' Description: ' Procedures : ConsDataExists3() ' ConsImport3() ' OpenImport3() ' OpenReconciliation3()
	ClearForecastData	175 - 196	
	ConsDataExists3	23 - 41	'+ ' Procedure : ConsDataExists3 ' Comments : checks if data exists in import tables ' Parameters: - ' Returns : Boolean - ' Modified : 29 Sep 2006 MAT ' -

ModuleName	Procedure	Lines	Comments
	ConsImport3	51 - 91	
	OpenImport3	101 - 132	
	OpenReconciliation3	142 - 167	
basXMLProductImport	(General Declarations)	1 - 15	'+ ' Module : basXMLProductImport ' Description: ' Procedures : ConsDataExists2() ' ConsImport2() ' OpenImport2() ' OpenReconciliation2()
	ClearSCMSData	151 - 178	
	ConsDataExists2	23 - 43	'+ ' Procedure : ConsDataExists2 ' Comments : checks if data exists in import tables ' Parameters: - ' Returns : Boolean - ' Modified : 24 Jul 2006 MAT ' -
	ConsImport2	53 - 83	
	InSubList	186 - 210	' -- ' Creates a delimited string of values, converts ' it to an array and determines if the array ' contents contain the specified string ' ==
	OpenImport2	93 - 116	
	OpenReconciliation2	126 - 143	
basXMLShipmentImport	(General Declarations)	1 - 16	'+ ' Module : basXMLShipmentImport ' Description: ' Procedures : ShipDataExists() ' ShipImport() ' OpenImportShipment() ' OpenReconciliationShipment()
	ClearShipmentData	153 - 181	
	OpenImportShipment	95 - 118	
	OpenReconciliationShipment	128 - 145	
	ShipDataExists	24 - 45	'+ ' Procedure : ShipDataExists ' Comments : checks if data exists in import tables ' Parameters: - ' Returns : Boolean - ' Modified : 1 Sep 2009 mahmed ' -
	ShipImport	55 - 85	
clsCloneProduct	(General Declarations)	1 - 7	
	CloneConsumption	197 - 233	
	createCloneTable	126 - 184	
	ProduceConsumptionRecords	246 - 322	
	ProductID	36 - 47	'+ ' Property(S): ProductID ' Comments : ' Parameters : ' Created : 16 Jul

ModuleName	Procedure	Lines	Comments
			2009 jleiner ' Modified : '-
	ProductID	16 - 28	'+ ' Property(G): ProductID ' Comments : ' Parameters : ' Returns : String ' Created : 16 Jul 2009 jleiner ' Modified : '-
	setStartDate	99 - 110	
	StartDate	77 - 88	
	StartDate	57 - 67	
clsCloseCommand	(General Declarations)	1 - 46	'+ ' Module : clsCloseCommand ' Description: Control the Access close button. ' When previewing reports, it is very easy for the user to close PipeLine while ' only attempting to close the previewed report and return to PipeLine. ' This occurs because the user confuses the Red X button in the upper ' right corner of the screen for the command "close report" ' when it actually executes the command "exit PipeLine".
	Enabled	64 - 78	
	Enabled	47 - 62	
clsConsumptionDataXML	(General Declarations)	1 - 45	'+ ' Module : clsConsumptionDataXML ' Description: Object for importing consumption data in Supply Chain Manager ' : XML format ' Procedures : Import() ' LoadXML(strFile As String) ' Get DataInterval() ' Let DataInterval(ByVal dblValue As Double) ' Get DataSource() ' Let DataSource(ByVal strName As String) ' Get DateExported() ' Let DateExported(ByVal dtmValue As Date) ' Get EndPeriod() ' Let EndPeriod(ByVal dtmValue As Date) ' Get StartPeriod() ' Let StartPeriod(ByVal dtmValue As Date) ' Get SystemName() ' Let SystemName(ByVal strName As String)
	DataInterval	260 - 272	
	DataInterval	239 - 251	
	Datasource	303 - 315	
	Datasource	282 - 294	
	DateExported	346 - 358	
	DateExported	325 - 337	
	EndPeriod	389 - 401	
	EndPeriod	368 - 380	
	Import	46 - 155	

ModuleName	Procedure	Lines	Comments
	LoadXML	167 - 229	
	StartPeriod	432 - 444	
	StartPeriod	411 - 423	
	SystemName	454 - 466	
	SystemName	475 - 487	
clsExportExcel	(General Declarations)	1 - 46	
	Class_Initialize	553 - 556	
	Class_Terminate	557 - 568	
	CloseFile	128 - 133	
	ColumnLetter	260 - 279	
	CreateFile	65 - 90	
	CreateWorksheet	143 - 149	
	deleterows	117 - 120	
	FormatCells	300 - 359	
	FormatSelection	445 - 485	
	FormatWorksheet	214 - 229	
	getcolumn	282 - 285	
	getexcelfilename	47 - 64	
	GetLastActiveColumn	368 - 375	
	GetLastActiveRow	360 - 367	
	GetValue	288 - 290	
	InsertRows	376 - 381	
	MergeCells	291 - 299	
	MoveActiveSheetToEnd	538 - 540	
	OpenFile	121 - 127	
	PageSetup	396 - 444	
	PutValueN	105 - 116	
	PutValueS	100 - 104	

ModuleName	Procedure	Lines	Comments
	RemoveEmptySheets	542 - 552	
	RemoveEqualValues	486 - 537	
	RemoveWorksheets	150 - 156	
	RenameWorkSheet	157 - 209	
	SaveFile	135 - 140	
	SelectRange	382 - 395	
	SelectWorksheet	210 - 213	
	setcolumn	257 - 259	
	setcolwidth	91 - 95	
	setrowcol	96 - 99	
	SetValue	230 - 256	
clsForecastDataXML	(General Declarations)	1 - 50	'+ ' Module : clsForecastDataXML ' Description: Object for importing consumption data from Quantimed/SCMS ' : XML format ' Procedures : Import() ' LoadXML2(strFile As String) ' Get DataInterval() ' Let DataInterval(ByVal dblValue As Double) ' Get DataSource() ' Let DataSource(ByVal strName As String) ' Get DateExported() ' Let DateExported(ByVal dtmValue As Date) ' Get EndPeriod() ' Let EndPeriod(ByVal dtmValue As Date) ' Get StartPeriod() ' Let StartPeriod(ByVal dtmValue As Date) ' Get SystemName() ' Let SystemName(ByVal strName As String)
	DataInterval	407 - 419	
	DataInterval	428 - 440	
	Datasource	450 - 462	
	Datasource	471 - 483	
	DateExported	493 - 505	
	DateExported	514 - 526	
	EndPeriod	557 - 569	
	EndPeriod	536 - 548	
	Import	51 - 227	
	LoadXML2	239 - 397	

ModuleName	Procedure	Lines	Comments
	StartPeriod	579 - 591	
	StartPeriod	600 - 612	
	SystemName	622 - 634	
	SystemName	643 - 655	
clsProductDataXML	(General Declarations)	1 - 34	'+ ' Module : clsProductDataXML ' Description: Object for importing product data from SCMS source into PL ' : XML format ' Procedures : Import() ' LoadXML(strFile As String)
	Import	35 - 410	
	ImportCountries	600 - 639	
	LoadXML	422 - 598	
clsShipmentDataXML	(General Declarations)	1 - 28	'+ ' Module : clsShipmentDataXML ' Description: Object for importing product data from SCMS source into PL ' : XML format ' Procedures : Import() ' LoadXML(strFile As String)
	checkDuplicateMapping	533 - 581	
	Import	29 - 98	
	ImportDatasources	233 - 272	
	ImportFundingsources	274 - 313	
	ImportProducts	315 - 379	
	ImportShipments	382 - 527	
	ImportSuppliers	182 - 231	
	LoadXML	108 - 180	
modEstimation	(General Declarations)	1 - 15	'+ ' Module : modEstimation ' Description: ' Procedures : creaMonthlyTrendConsumption(strProductID As String, intEndYear As Integer) ' GetPeriodAverages(dtmStart As String, dtmEnd As String, blnUseEstimates As Integer) ' TempAttach(strDatabase)
	creaMonthlyTrendConsumption	32 - 300	'+ ' Procedure : creaMonthlyTrendConsumption ' Comments : Create MonthlyConsumption table for strProductID, ' : ending in intEndYear. First creates two temporary files from ' : data in table Consumption. One file contains monthly estimates ' : for actual consumption. One file contains monthly estimates for ' : forecast consumption. The monthly consumption is the amount ' : consumed averaged over the number of months; the last month in ' : the period absorbs any rounding errors. Finally, merge the two ' :

ModuleName	Procedure	Lines	Comments
			files, with actual consumption, where available, taking precedence ' : over forecast. ' Parameters: strProductID ' intEndYear - ' Returns : - ' Modified : 01 Jul 1998 JSL ' -
	GetPeriodAverages	314 - 544	
	GetPeriodAveragesForecast	557 - 789	
	TempAttach	799 - 823	
modHelpContextIDs	(General Declarations)	1 - 13	' Module : modHelpContextIDs ' Description: ' Procedures : CreateHelpContextIDs() ' CreateHelpContextIDsReports() ' GetHelpID(strForm As String, fIsForm As Boolean, bytIndex As Byte)
	CreateHelpContextIDs	14 - 55	' Comments : Generate a table of the HelpContext IDs text for all form ' : Objects ' Parameters : None ' Returns : None ' Created : 03/23/2000 JSL '---
	CreateHelpContextIDsReports	57 - 99	' Comments : Generate a table of the HelpContext IDs text for all Report ' : Objects ' Parameters : None ' Returns : None ' Created : 03/23/2000 JSL '---
	GetHelpID	101 - 156	' Comments : ' Parameters : ' Returns : String ' Created : 02-Aug-00 Jeff Leiner ' Modified : ' ' --
modInterpolation	(General Declarations)	1 - 14	'+ ' Module : modEstimation ' Description: ' Procedures : creaMonthlyTrendConsumption(strProductID As String, intEndYear As Integer) ' GetPeriodAverages(dtmStart As String, dtmEnd As String, blnUseEstimates As Integer) ' TempAttach(strDatabase)
	Interpolate	15 - 59	
Table Initialization	(General Declarations)	1 - 26	'+ ' Module : Table Initialization ' Description: ' Procedures : AttachData(intStartup As Integer) ' AttachProgram() ' CheckProductDSLMonths() ' CheckY2kDates() ' CopyFile(strSource As String, strDestination As String) ' CopyFile_TSB(strSource As String, strDestination As String) ' GetAttachedPath(strDatabase As String, strTable As String) ' GetDBUpdateDate() ' iniTableLang() ' KillFile_tsb(strFile As String) ' RemoveOldVersion() ' SetLanguage(strLanguage As String) ' StartUp() ' UpdateShipData()
	AttachData	43 - 308	'+ ' Procedure : AttachData ' Comments : ' Update connection information to reflect FP program that is active, ' as indicated by IsCurrent setting in table Program. ' If intStartUp is true, error returns invoke form frmProgram, ' which allows the user to reconfigure. ' If new tables are added to the program, or attached, side of the tables, ' they must be added to this

ModuleName	Procedure	Lines	Comments
			routine. ' Tables used to hold results for display of printing are cleared, ' to eliminate any references to installation specific lookup variables ' Parameters: intStartup - ' Returns : Integer - ' Modified : 01 Jul 1998 JSL '12 May 2004 LKB ' -
	AttachProgram	319 - 366	
	CheckProductDSLMonths	377 - 426	
	CheckY2kDates	439 - 503	
	CopyFile	514 - 529	
	CopyFile_TSB	540 - 582	
	GetAttachedPath	593 - 625	
	GetDBUpdateDate	636 - 652	
	iniTableLang	663 - 843	
	KillFile_tsb	853 - 868	
	RemoveOldVersion	878 - 900	
	SetLanguage	911 - 980	
	StartUp	990 - 1022	
	UpdateShipData	1032 - 1047	
Table Utils	(General Declarations)	1 - 34	'+ ' Module : Table Utils ' Description: ' Procedures : AtNewRecord(frm As Form) ' CodeUsed(strCode As String, strTable As String, strCodeField As String) ' CreateMonthlyAction(intEndYear As Integer) ' CreateMonthlyConsumption(strProductID As String, intEndYear As Integer, intKeepData As Integer) ' CreateMonthlyMethodConsumption(varStrMethod, varIntEndYear) ' CreateMonthlyMethodStock(strMethod As String, intEndYear As Integer, Optional fSkipPlanned As Boolean) ' CreateMonthlyProblem(intEndYear As Integer) ' CreateMonthlyShipment(strProductID As String, Optional fSkipPlanned As Boolean) ' CreateMonthlyStock(strProductID As String, ByVal intEndYear As Integer, intKeepData As Integer, Optional fSkipPlanned As Boolean) ' CreateShipSched(intEndYear As Integer) ' GetDesLevel(varProductID) ' GetDesMin(varProductID) ' GetLastYear(strProductID As String) ' getLeadTime(strProductID As String, strSupplierID As String, intLTType As Integer) ' getmaxMonths(strProductID

ModuleName	Procedure	Lines	Comments
			As String) ' GetMethodActualorForecast(intYear As Integer, strPeriod As String, intMonth As Integer) ' getminMonths(strProductID As String) ' getParmValue(varParm) ' SetCentury(vardate) ' SetParm(strPArm As String, strValue As String)
	AtNewRecord	44 - 57	'+ ' Procedure : AtNewRecord ' Comments : Returns true if record set in frmForm is positioned at new record ' : Reads bookmark; if recordset is at new record, Access generates an error return ' Parameters: frm - ' Returns : - ' Modified : 01 Jul 1998 JSL ' 20 Jun 2003 LBlanken ' -
	CodeUsed	71 - 126	
	CreateMonthlyAction	144 - 371	
	CreateMonthlyConsumption	391 - 716	
	CreateMonthlyMethodConsumption	726 - 775	
	CreateMonthlyMethodStock	788 - 880	
	CreateMonthlyProblem	896 - 1103	
	CreateMonthlyShipment	1120 - 1219	
	CreateMonthlyStock	1240 - 1663	
	CreateShipSched	1681 - 1874	
	GetDesLevel	1887 - 1918	
	GetDesMin	1931 - 1983	
	GetLastYear	1996 - 2048	
	getLeadTime	2067 - 2136	
	getmaxMonths	2148 - 2169	
	GetMethodActualorForecast	2183 - 2247	

ModuleName	Procedure	Lines	Comments
	getminMonths	2259 - 2280	
	getParmValue	2291 - 2311	
	GetProductID	2321 - 2333	
	SetCentury	2345 - 2399	
	SetParm	2412 - 2436	

Menu Structure

Overview

PipeLine contains a dynamic menu structure. The menu is created each time the application is open and when the language is changed. The code that is ran to create the menu bar can be found in AddNewMB which calls the LangMenus in the basToolbarFunctions module. This code sets the OnAction property for the menu item along with the help file and context ID. In some cases, the OnAction code calls other modules and in other cases it calls macros. The following table lists the menu item and the OnAction command it calls.

Program calls by Menu choices

Table 20 - Program Calls by Menu Choices

Menu Option	Sub Menu		OnAction command	Description
File	New		=FileMenu(""new"")	Click to create New PipeLine Data files
	Open		=FileMenu(""Open"")	Click to open existing PipeLine Data file
	Copy		=FileMenu(""Copy"")	Click to copy current data file to a new location
	Close		=FileMenu(""Close"")	Deactivate current data file
	Properties		=OpenProperties()	Open properties form
	Exit		=FileMenu(""Exit"")	Exit PipeLine
Import	Products	New	mcrProduct.ImportNew	Import new product file
		Update	mcrProduct.ImportUpdate	Updates existing products that match files in product file
	Consumption	Forecast	mcrConsumption.ImportForecast	Import forecast data from Quantimed
		Actual	mcrConsumption.Import	Imports actual data from Supply Chain Manager
		Reconcile	mcrConsumption.Reconcile	Reconciles last Supply Chain Manager import
	Shipments	Initial	mcrShipments.ImportPipleLine	
		PipeLine	mcrMenu.ToolsImportShipments	Import shipments
Export	Program Data		mcrMenu.ExportData	Exports all data
	Shipments		mcrMenu.ExportData	Exports all shipment data
Tools	Language	English	mcrmenu.LanguageE	Changes display language to English
		Português	mcrmenu.LanguageP	Changes display language to Portuguese
		Français	mcrmenu.LanguageF	Changes display language to French
		Español	mcrmenu.LanguageS	Changes display language

Menu Option	Sub Menu		OnAction command	Description
				to Spanish
Tools *(Cont.)*	PipeLine Summary		mcrMenu.PLSummary	Opens PipeLine Summary
	Compact Backend		mcrMenu.CompactBE	Compacts current data file
	Choose PDF Printer		mcrmenu.ChoosePDF	Opens form to select PDF printer
Window	*<various>*		=ChangeProgram("' & rstPrograms!ProgramName & '")	Opens activated program files
Help	Help		=OpenHelp()	Opens help file
	About PipeLine		=OpenAbout()	Opens PipeLine about form

Archiving and Backup

Overview

Currently there is not an archiving function built in to PipeLine. One of the benefits however is that the backend database can be copied and saved at any time (better to be done while PipeLine is closed). Users can also utilize the copy function found in the File menu to easily copy the backend file to another filename for backup purposes.

It is also recommended that the backend database file compacted and repaired at regular intervals.

Interfaces with Other Systems

Overview

PipeLine allows the user to interface with generic systems through the export of all program data, export of shipment data, and import of product, consumption, and shipment data via an xml file format.

Generic systems (Export of all program data)

PipeLine can interface with generic systems through the exporting of an xml file. The xml is expected to follow a basic schema. The schema is explained in the PipelineXMLOutputSchema_070924.xsd file found on the developer source code cd in the /schemas folder.

Figure 51 - xml Export file schema diagram

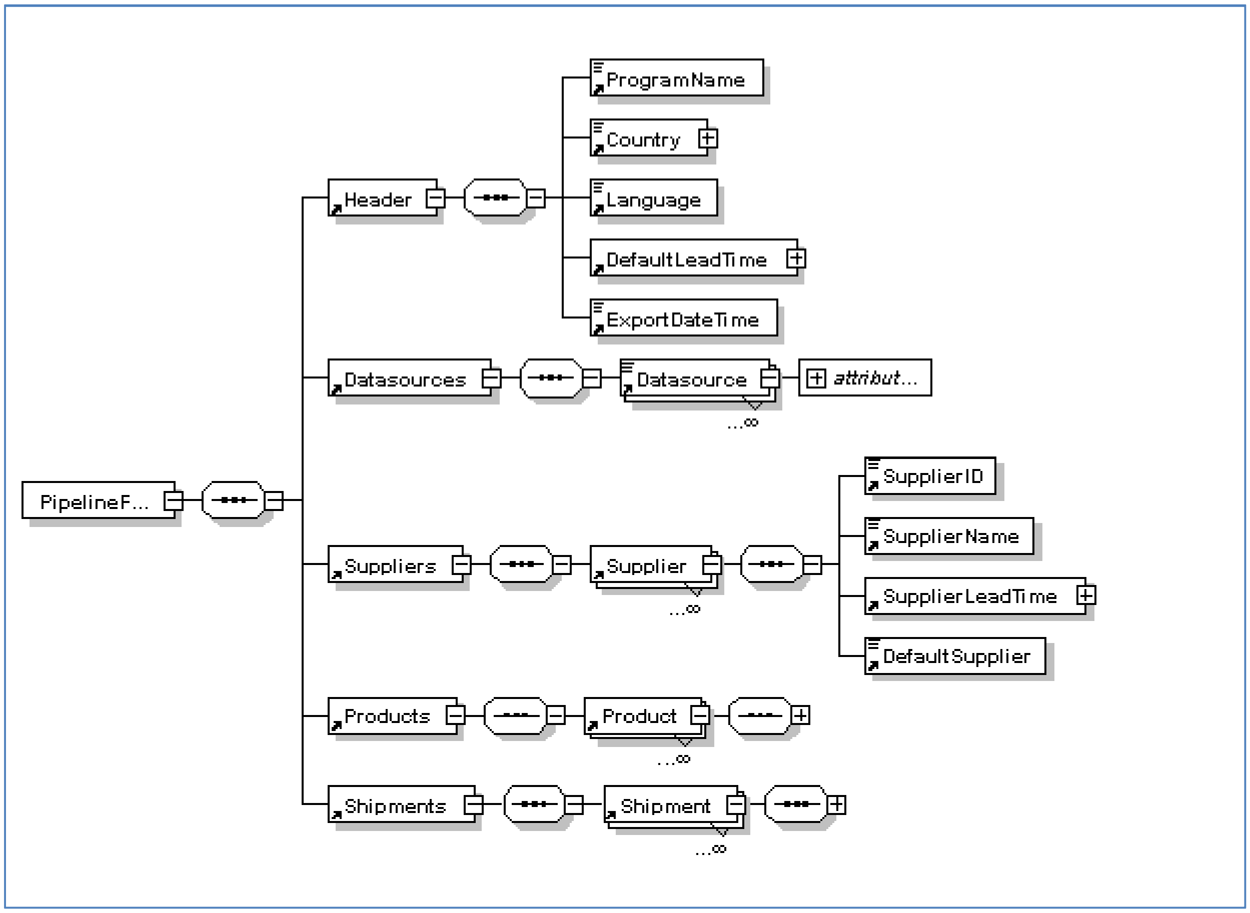


Figure 52 - xml Export file schema diagram

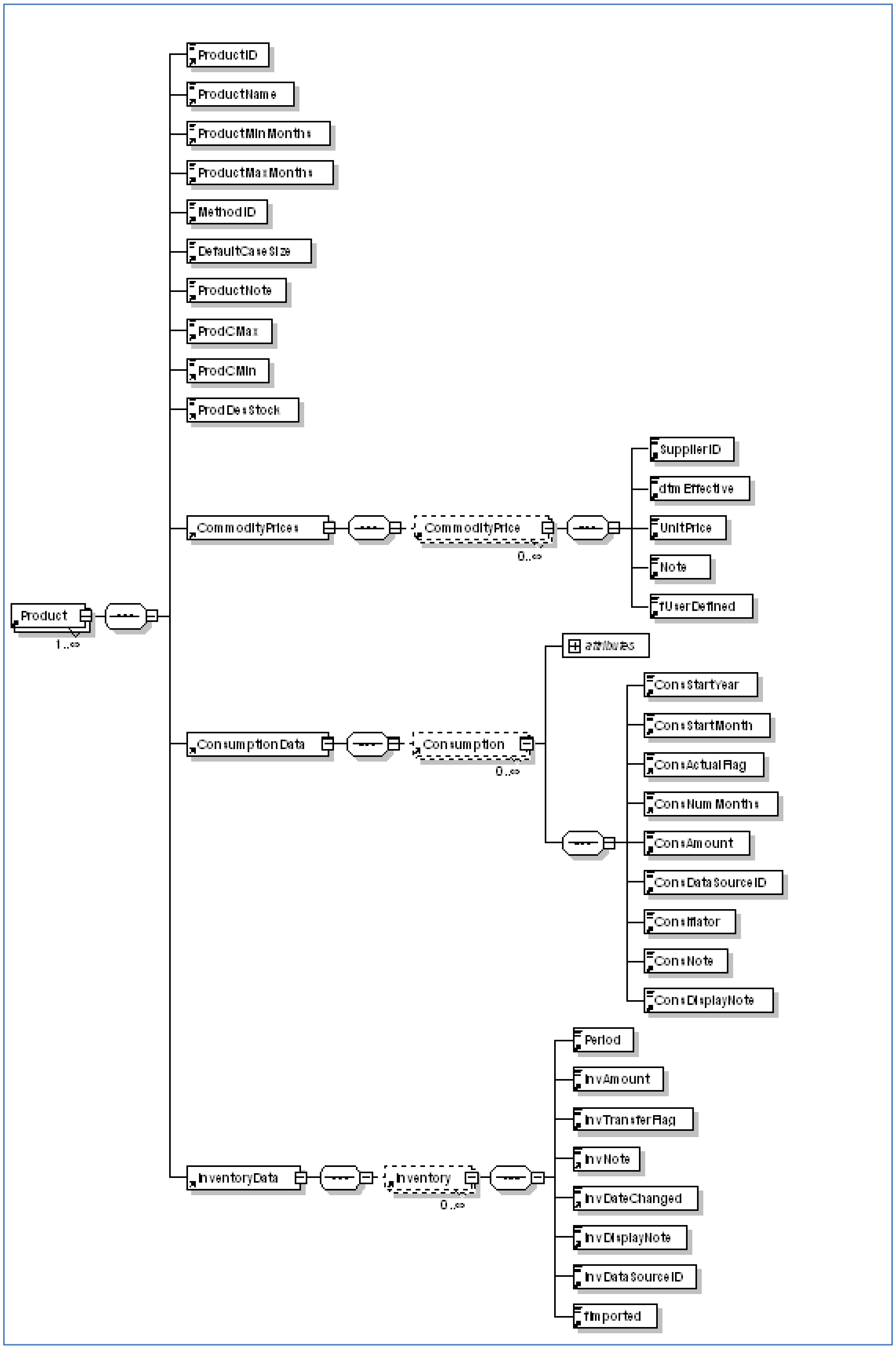
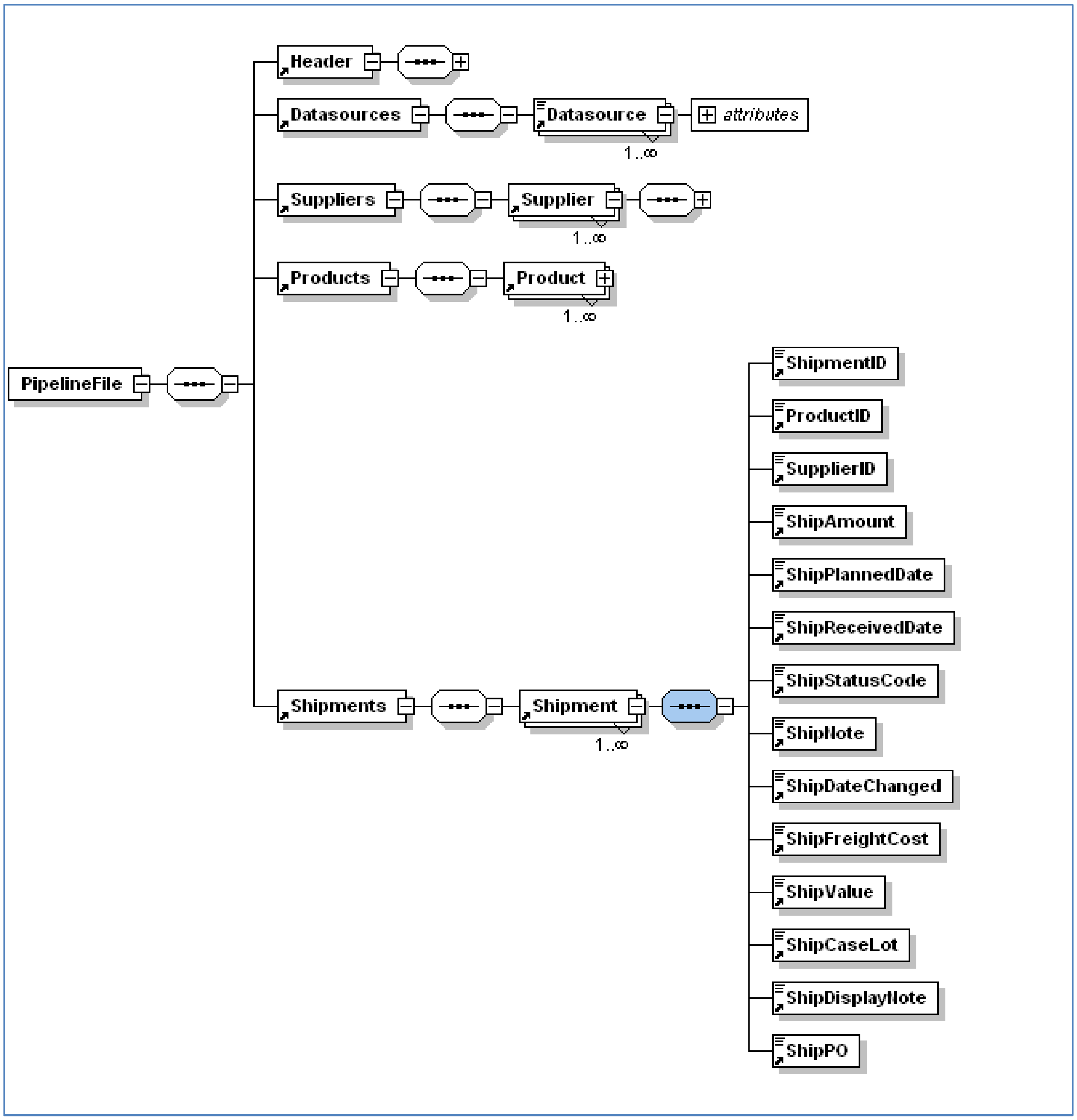


Figure 53 - xml Export file schema diagram



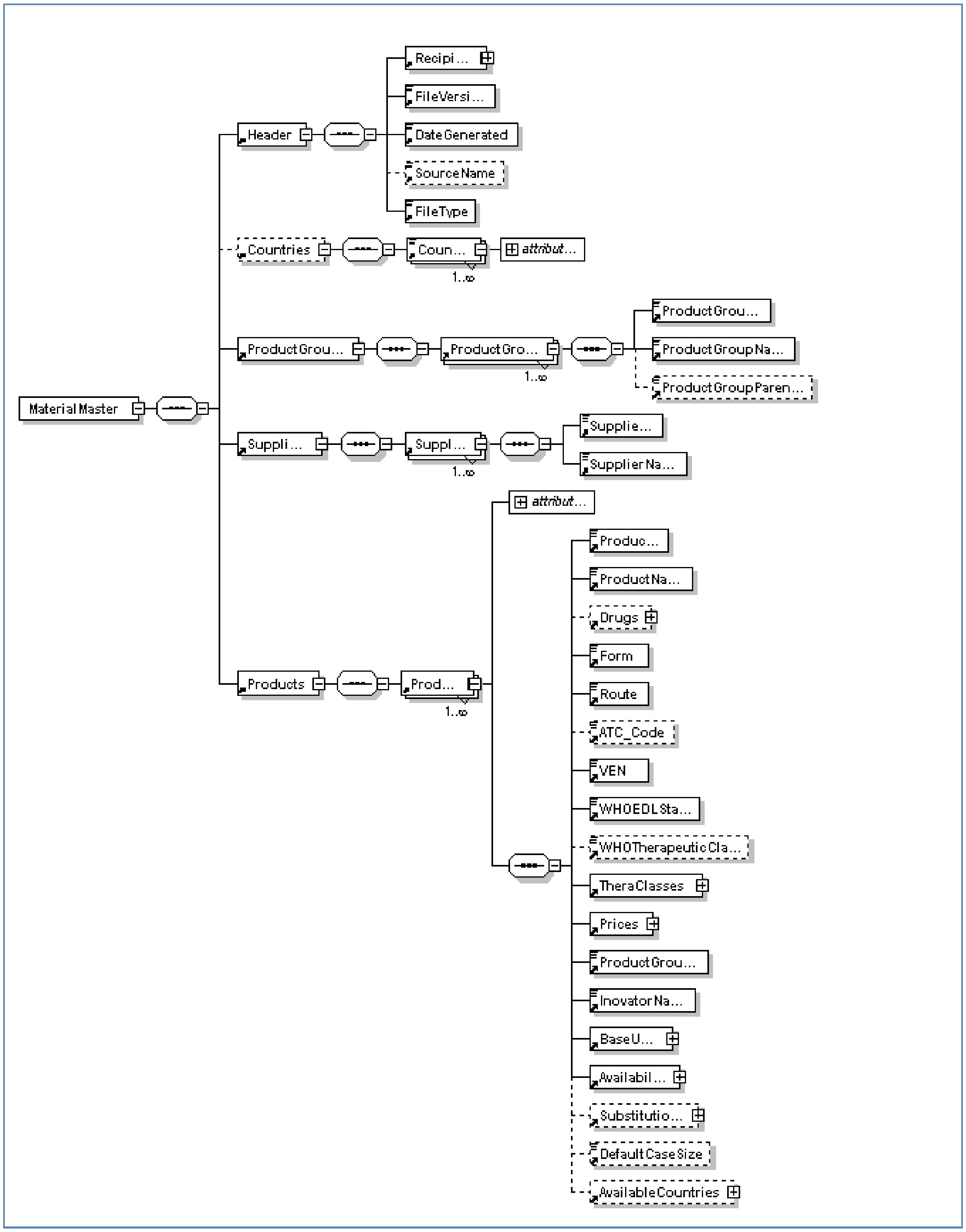
Other Pipeline Databases (import shipment data)

PipeLine can interface with other PipeLine database through the importing of an xml file. The xml is expected to follow the schema outlined for generic system above.

E-Catelog (Import product data)

PipeLine can interface with E-Catelog (or other product list sources) through the importing of an xml file. The xml is expected to follow a basic schema. The schema is explained in the MaterialMasterNorm_070516_NoCustomType.xsd file found on the developer source code cd in the /schemas folder.

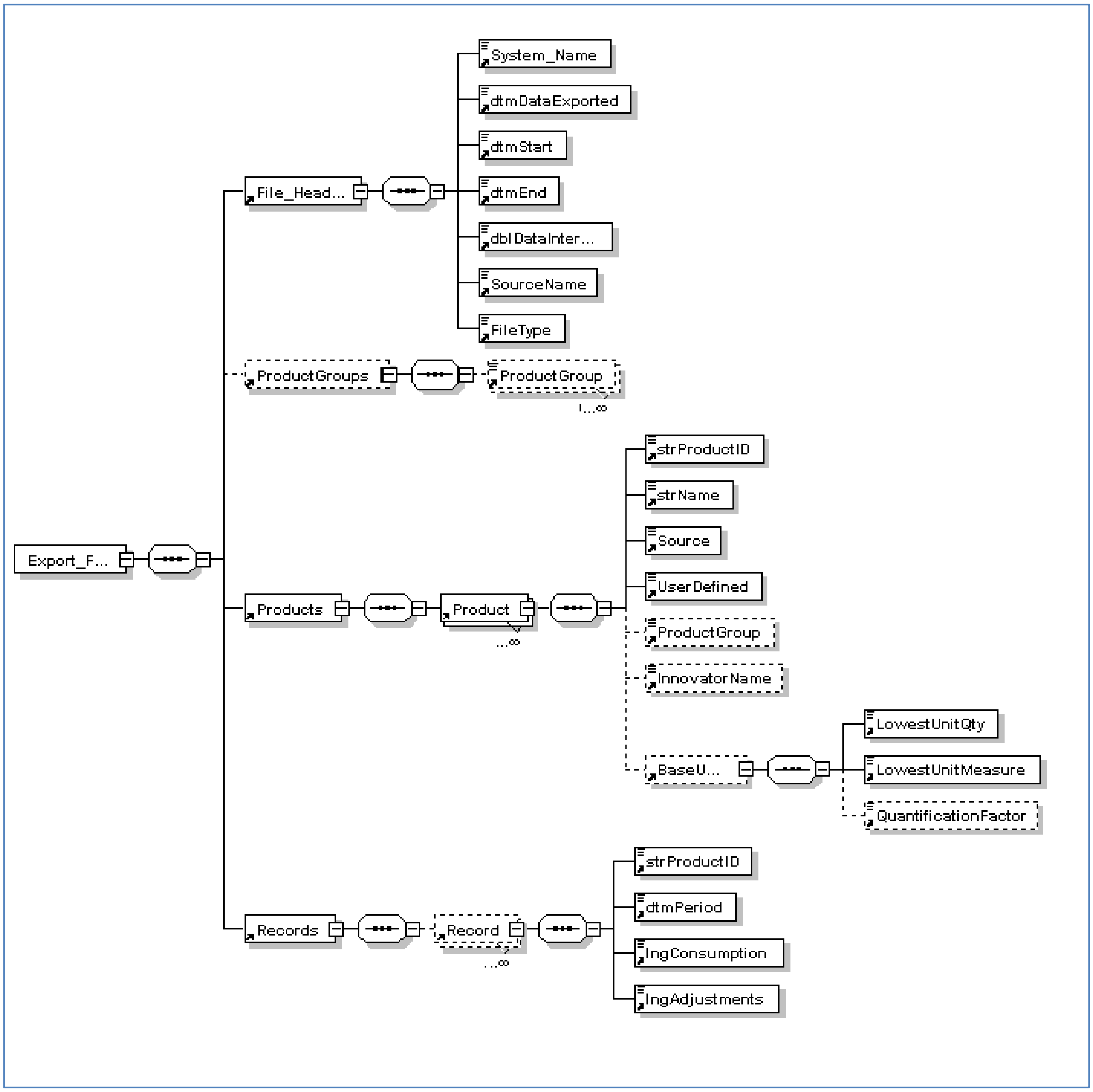
Figure 54 - xml Export file schema diagram



Quantimed (import forecast consumption data)

PipeLine can interface with Quantimed through the importing of an xml file. The xml is expected to follow a basic schema. The schema is explained in the QuantimedForecastOutput.xsd file found on the developer source code cd in the /schemas folder.

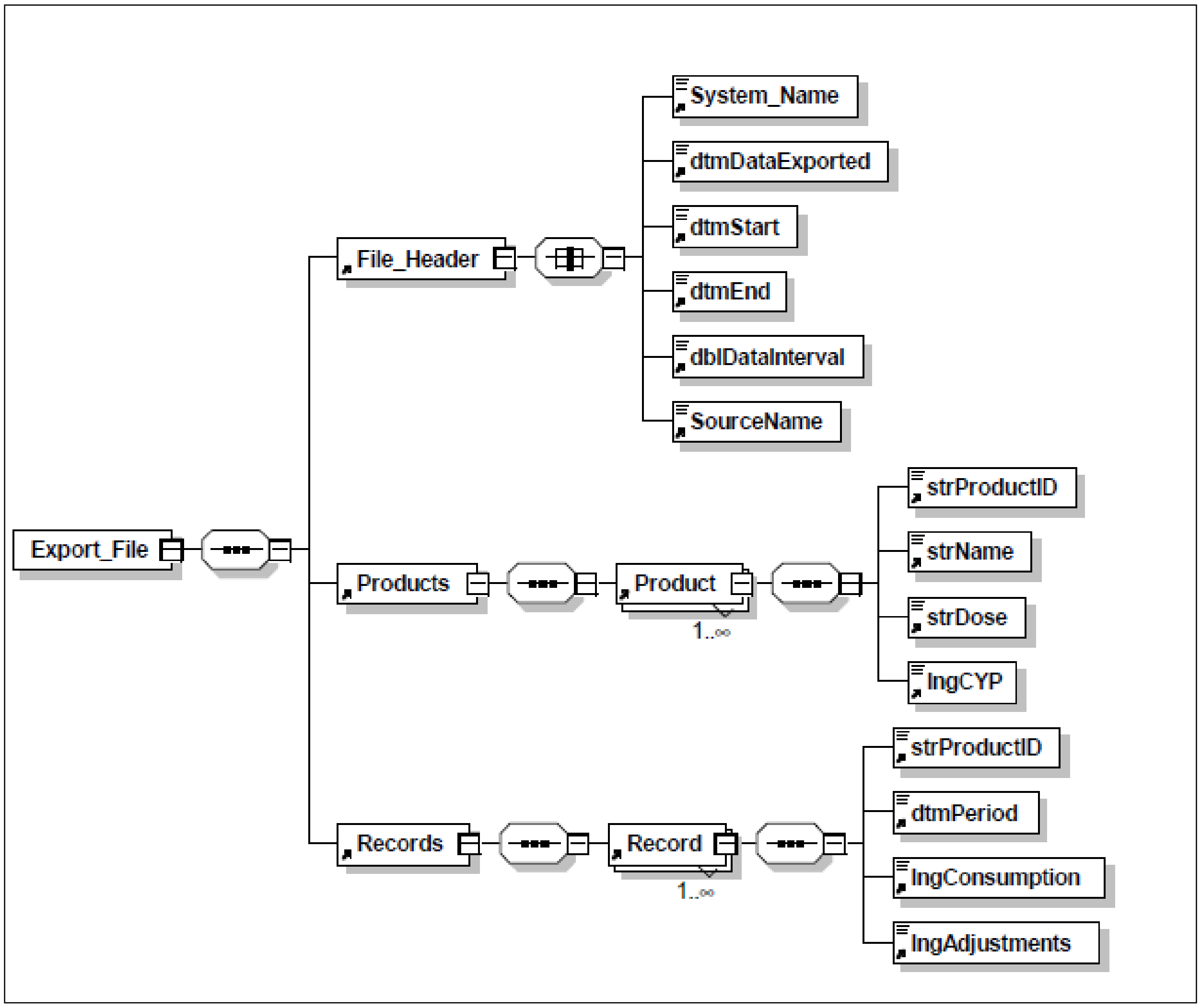
Figure 55 - xml Export file schema diagram



Supply Chain Manager (import actual consumption data)

PipeLine can interface with Supply Chain Manager through the importing of an xml file. The xml is expected to follow a basic schema. The schema is explained in the SCMgr_PipeLine_Export.xsd file found on the developer source code cd in the /schemas folder.

Figure 56: xml Export file schema diagram



APPENDIX A: UPGRADING DATA FILES

Overview

When upgrading PipeLine sometimes changes need to occur in the backend data files. In PipeLine a mechanism has been created that will easily allow the system to do this automatically when opening a previous versions file. Basically, the application will determine which version the data file is, will create a new file as specified by the user, and then run the appropriate SQL statements to copy the data into the file.

Version Numbering

PMP_MPTY.MDB is the empty datafile for PipeLine. When creating a new backend, PipeLine copies this file to the specified location and adds the program data to it. The table, tblBE_Version, in this file contains the version number of the backend along with the date that the file was updated. When PipeLine opens a new datafile, it verifies this number with the number stored in the backend that it is opening. (See code: CheckBackEndVersion). If the version numbers are different then the upgrade procedure is called. If the numbers are the same then the file will open and the user is taken to the main menu. Please note that when making structure changes to the data is is imperative that the changes be made in the the PMP_MPTY.MDB file and that this version number is updated.

Upgrade Procedure

When the version number in the backend that the user is opening does not match the number in the PMP_MPTY.MDB file, the upgrade procedure is called (see code: UpgradeProgram). This procedure prompts the user for a new file name and location for the upgraded datafile. It then creates a log file to keep track of all changes and errors that occur during the upgrade. The procedure will copy the PMP_MPTY.MDB to location specified and then opens the table tlkUpgrade and runs the appropriate sql statement. Each statement is created to copy the data from the existing backend into the new backend. This ensures that all the data gets placed in the correct location in the new backend. Once all the SQL statements are run, the procedure will change the name to the one specified and open the file. The user is then taken to the Main Menu. Please note again, that the original datafile remains intact during this procedure. This guarantees the user will not lose any data.

SQL Statements Used

The most complicated part of the upgrade procedure is writing the SQL statements. There are three types of SQL statements to be written. They are: Update, Append and Delete. Each table should have at least one of each of these but in some cases can have multiple Append and Update statements. In order to determine which statements to run, a version number is stored with each statement along with an order number. The order number specified the order in which the SQL statements are ran, and the version number dictates which code is ran depending on the version of the existing data backend. When updating the version number, the new upgrade version number is double the previous number. (For example, version 3.08 has a version number of 64. Version 3.09's number would then be 128.) This ensures that numbers will not duplicate. To specify, which versions the SQL statement is ran for, the version number in the table is the sum of the version number is runs for. (And so, number 129 would run when the backend is version 3.09 AND 3.01.) With each release, the version numbers in the table must be updated along with each statement. The upgrade procedure will then determine which SQL statements to run based on this numbering system and the verstion number of the existing data. The delete statement is ran first so that non system data is deleted. The update statements are next. These will match any remaining system data and update as needed. And finally, the append statements will then copy in all other existing data. This procedure ensures that all data is updated to the new structure correctly.

APPENDIX B: PIPELINE SUMMARY

Overview

This option (only available in English) is intended for program managers and/or consultants responsible for managing multiple programs tracked by PipeLine. It aggregates PipeLine data from selected programs, and arranges the data so it can be presented in a selected graph or report.

The following graphs and reports are available:

Shipment Costs by Supplier Report
This report is similar to the PipeLine Shipment Summary report. It displays shipment costs, including product and freight costs, for selected supplier/status of selected, aggregated programs.

Shipment Orders Report
This report is similar to the PipeLine Shipment Orders report. It displays quantities and cost of orders for a selected supplier/status of selected, aggregated programs.

Consumption Graph/Export
This graph is similar to the PipeLine Consumption graph. It produces a monthly or quarterly bar chart showing actual and forecast consumption by product or method of selected, aggregated programs. This option lets you export aggregated data.

Couple-Years of Protection (CYP) Graph
This graph is similar to the PipeLine couple-years of protection (CYP) graph. It produces a monthly or quarterly bar chart showing actual and forecast CYP by product or method of selected, aggregated programs.

Access the PipeLine Summary Module

The PipeLine Summary Module is accessed through PipeLine.

From the Menu Bar—

1. Click on the Tools option of the Menu Bar to display the Tools pull-down menu.
2. Click on the PipeLine Summary pull-down menu option to display the PipeLine Summary menu screen.

Selecting Programs

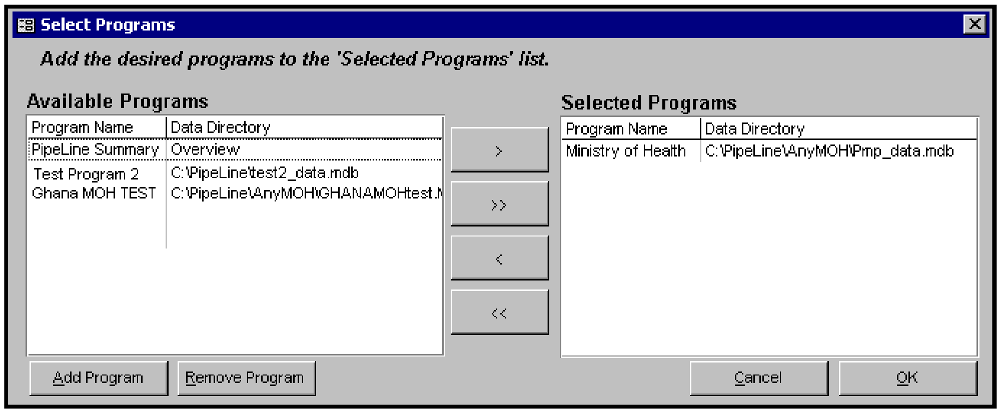
The Select Programs option lets you select the program(s) to aggregate. You must select a program or programs for aggregation before you create reports or graphs.

From the PipeLine Summary Menu—

1. Click on the Select Programs button.

PipeLine displays the Select Program screen.

Figure 57 - Select Program



Program Name	Data Directory
PipeLine Summary	Overview
Test Program 2	C:\PipeLine\test2_data.mdb
Ghana MOH TEST	C:\PipeLine\AnyMOH\GHANAMOHtest.M

Program Name	Data Directory
Ministry of Health	C:\PipeLine\AnyMOH\Pmp_data.mdb

If the program (or programs) you need to aggregate does not appear in the list of available programs, you can add them.

Adding Programs

Use the Add Program button to add programs for selection and aggregation.

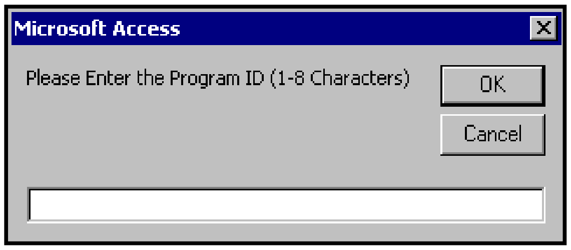
1. Click on the Add Programs button

PipeLine displays the Load Database window.

2. Locate the data file corresponding to the program you want to add.
3. Click on the program you need, then click on the Open button.

PipeLine displays a window so you can associate an eight-character ID with the selected program.

Figure 58 - Enter Program ID window



Please Enter the Program ID (1-8 Characters)

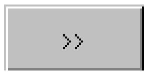
4. Type the program's eight-character ID.
5. Click on the OK button to save the ID, and close the Program ID window.

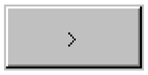
The selected program is displayed in the Available Programs window, on the Select Program screen.

6. Repeat the process until you have added all the programs you need.

After your list is complete, you must select the programs you need to aggregate for reporting. This can be done in two ways:

7. Click on the Select All button to move the programs in the Available Programs window to the Selected Programs window
   Or, click on the program you need, and click on the Select button to move it from the Available Programs window to the Selected Program window.
8. Repeat, as needed.





After you select the programs you need—

9. Click on the OK button to aggregate the data and return to the PipeLine Summary menu.

Changing Data
If you make changes to a program's data in PipeLine, you must reload the program into PipeLine Summary to ensure that the changes are reflected in the summary reports and graphs.

Creating Reports and Graphs

PipeLine can create summary reports and graphs.

Shipment Costs by Supplier Report

The Shipment Costs by Supplier report groups shipment quantities and costs by supplier, method, product, program, and status.

This is useful if you have shipments going to multiple programs from the same supplier, and you need to look at total shipments and associated costs for each supplier.

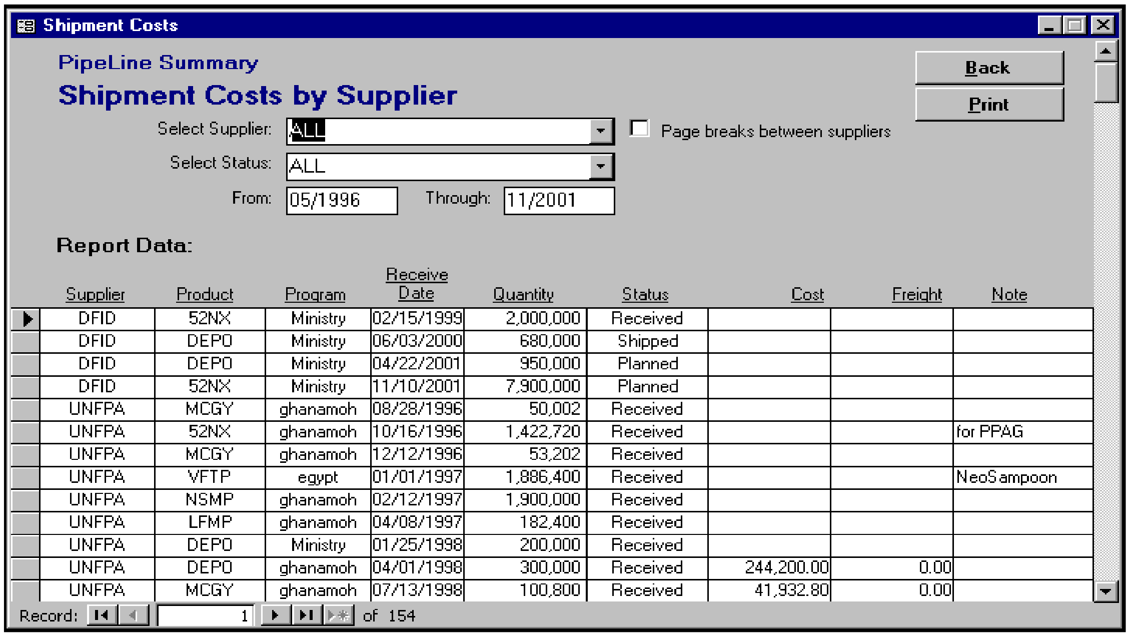
Use this report to review shipment information (including schedules and budgets) for a selected set of programs.

From the PipeLine Summary Menu—

1. Click on the Shipments by Supplier Report button.

PipeLine displays shipments and their associated costs. The Shipment Costs by Supplier screen is displayed after the report is created. Use this screen to create shipment cost reports for other suppliers.

Figure 59 - — PipeLine Summary Shipment Costs by Supplier screen



PipeLine Summary

Shipment Costs by Supplier

Select Supplier: ALL

☐ Page breaks between suppliers

Select Status: ALL

From: 05/1996 Through: 11/2001

Report Data:

Supplier	Product	Program	Receive Date	Quantity	Status	Cost	Freight	Note
DFID	52NX	Ministry	02/15/1999	2,000,000	Received			
DFID	DEPO	Ministry	06/03/2000	680,000	Shipped			
DFID	DEPO	Ministry	04/22/2001	950,000	Planned			
DFID	52NX	Ministry	11/10/2001	7,900,000	Planned			
UNFPA	MCGY	ghanamoh	08/28/1996	50,002	Received			
UNFPA	52NX	ghanamoh	10/16/1996	1,422,720	Received			for PPAG
UNFPA	MCGY	ghanamoh	12/12/1996	53,202	Received			
UNFPA	VFTP	egypt	01/01/1997	1,886,400	Received			NeoSampoon
UNFPA	NSMP	ghanamoh	02/12/1997	1,900,000	Received			
UNFPA	LFMP	ghanamoh	04/08/1997	182,400	Received			
UNFPA	DEPO	Ministry	01/25/1998	200,000	Received			
UNFPA	DEPO	ghanamoh	04/01/1998	300,000	Received	244,200.00	0.00	
UNFPA	MCGY	ghanamoh	07/13/1998	100,800	Received	41,932.80	0.00	

Record: 1 of 154

By default, the Shipment Costs by Supplier report is set for all products and all shipment statuses, with a reporting period of 5 years. To limit the report's scope—

2. Click on the arrow next to the Select Supplier field, and select a supplier from the pull-down menu.
3. Click on the Select Status field and select the shipment status you want.
4. Click on the From field, and type the beginning date of the report.
5. Click on the Through field, and type the ending date of the report.

The Page Breaks Between Suppliers fields let you include page breaks between suppliers listed in the report (when the report is generated for more than one supplier).

6. Click on the Page Breaks Between Suppliers field to include page breaks between suppliers.

Shipment Order Report

The Shipment Order report groups shipments by status, supplier, method, and product. Shipments with status *planned*" are grouped under "New shipments requested this order." Shipments with status *shipped* or status *ordered* are grouped under "Confirmation of shipments previously ordered and expected." Shipments with status *received* are grouped under "Shipments received (in report range)."

This is useful if you have shipments to multiple programs and you need to look at which shipments need to be ordered or confirmed with the suppliers or donors.

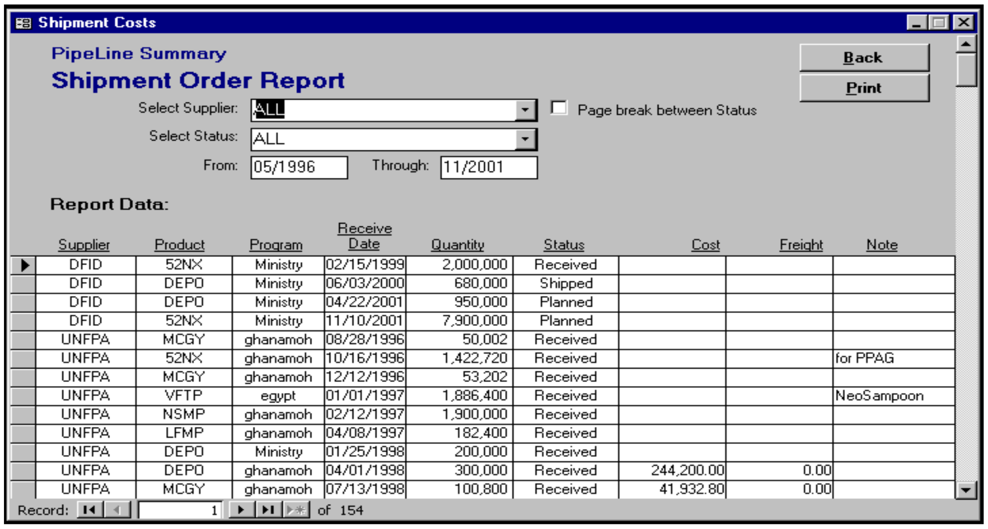
Use this report to review shipment status information (including schedules and costs) for a selected set of programs. Select a single supplier. Present this report to that supplier to order planned shipments, confirm previously ordered shipments, and confirm receipt of shipments received from the supplier during the report period.

From the PipeLine Summary Menu—

1. Click on the Shipment Order Report button.

PipeLine displays orders and associated data for the selected programs. By default, the Shipment Order report is set for all products and all shipment statuses with a 5-year reporting period.

Figure 60 - PipeLine Summary Shipment Order Report screen



Select Supplier: ALL

Page break between Status

Select Status: ALL

From: 05/1996 Through: 11/2001

Report Data:

Supplier	Product	Program	Receive Date	Quantity	Status	Cost	Freight	Note
DFID	52NX	Ministry	02/15/1999	2,000,000	Received			
DFID	DEPO	Ministry	06/03/2000	680,000	Shipped			
DFID	DEPO	Ministry	04/22/2001	950,000	Planned			
DFID	52NX	Ministry	11/10/2001	7,900,000	Planned			
UNFPA	MCGY	ghanamoh	08/28/1996	50,002	Received			
UNFPA	52NX	ghanamoh	10/16/1996	1,422,720	Received			for PPAG
UNFPA	MCGY	ghanamoh	12/12/1996	53,202	Received			
UNFPA	VFTP	egypt	01/01/1997	1,886,400	Received			NeoSampoon
UNFPA	NSMP	ghanamoh	02/12/1997	1,900,000	Received			
UNFPA	LFMP	ghanamoh	04/08/1997	182,400	Received			
UNFPA	DEPO	Ministry	01/25/1998	200,000	Received			
UNFPA	DEPO	ghanamoh	04/01/1998	300,000	Received	244,200.00	0.00	
UNFPA	MCGY	ghanamoh	07/13/1998	100,800	Received	41,932.80	0.00	

Record: 1 of 154

To limit the report's scope—

2. Click on the arrow next to the Select Supplier field, and select a supplier from the pull-down menu.
3. Click on the Select Status field, and select the shipment status you need.
4. Click on the From field, and type the beginning date of the report.
5. Click on the Through field, and type the ending date of the report.

The Page Breaks Between Status fields let you include page breaks between the shipment statuses listed in the report (when the report is generated for more than one shipment status).

6. Click on the Page Breaks Between Status field to include page breaks between shipment statuses.

Consumption Graph/Export

The Consumption graph groups consumption data by product ID or method ID.

Use this graph to compare relative shares and trends of distribution by product or method for multiple programs.

If you choose Method, the colors and legend will indicate what product and what program is represented by each Method ID bar. If you choose Method and Program Detail, you will see the relative shares of each program, but not product.

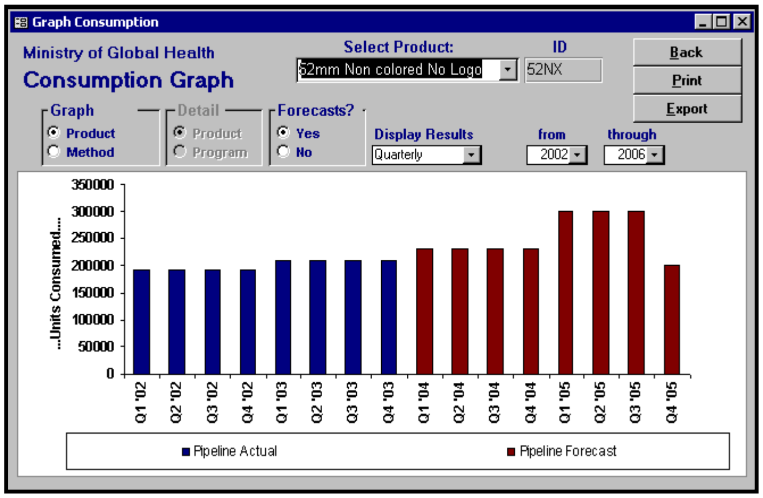
This is useful when programs use different, but comparable products, so the method distinction is more useful than the product distinction.

From the PipeLine Summary Menu—

1. Click on the Consumption Graph/Export button.

The graph is based on the last product/method and period updated or displayed. The Consumption Graph screen is displayed after the graph is created. Use this screen to create graphs for other products/methods.

Figure 61 - PipeLine Summary Consumption Graph screen



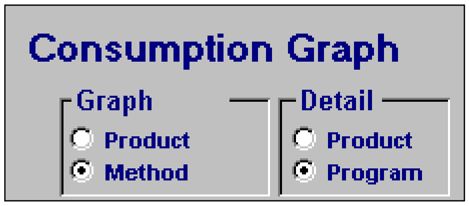
Graph Type

This section of the screen lets you determine the type of graph you will produce. Your options are Product or Method.

2. To create a graph by product, click on the arrow next to the Select Product field, and select a product from the pull-down menu.

   Or, click on the Method field. Click on the arrow next to the Select Method field, and select a method from the pull-down menu to create a graph by method.

Figure 62 - Graph options



When you create a consumption graph by method, the Detail section of the Consumption Graph screen becomes active. Product is selected by default, letting you create a graph for products associated with a selected method.

To create a graph for all products of a particular method through an aggregation of programs—

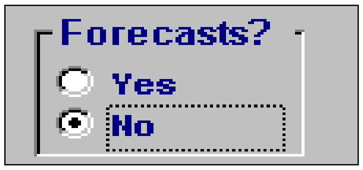
3. Click on the Program field.
4. Click on the arrow next to the Select Method field, and select a method from the pull-down menu.

Regardless of the type of graph you choose, select the remaining options.

By default, the graph includes consumption forecasts. To create a graph that excludes forecast data—

5. Click on the No field.

Figure 63 - Forecast options



Forecasts? Yes [✓ No]

6. Click on the arrow next to the Display Results field, and select Monthly to display the data by month or Quarterly to display the data by quarter.

The default reporting period is five years (the previous two years, current year, and following two years). To change the default reporting period—

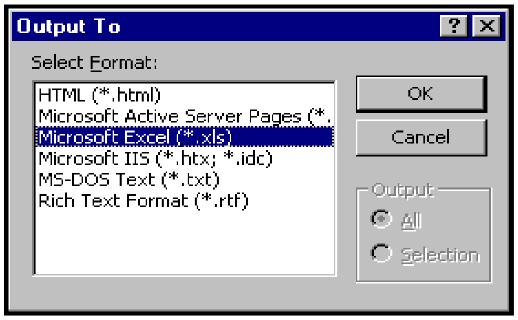
7. Click on the arrow next to the From field, and select the starting year of the report period.
8. Click on the arrow next to the Through field, and select the last year of the report period.

Exporting Summary Consumption Graph Data

Summary Consumption graph data can be exported for use with Microsoft Excel or other analysis software. After you create a consumption graph for a product or method—

1. Click on the Export button.

Figure 64 - Output window



Select Format:

HTML (*.html)
Microsoft Active Server Pages (*.
Microsoft Excel (*.xls)
Microsoft IIS (*.htx; *.idc)
MS-DOS Text (*.txt)
Rich Text Format (*.rtf)

PipeLine displays the Output window (see **Error! Reference source not found.**).

2. Click on the file type for the export data.
3. Click on the OK button.

PipeLine opens another Output window, so you can name and save the export file.

4. Click on the OK button.

   Or, click on the File Name field, and type the new export file name.

5. Click on the OK button to export and save the Consumption graph.
6. Click on the Back button to return to the PipeLine Summary menu.

Couple-Years of Protection (CYP) Graph

The CYP Graph converts consumption data by the CYP factor associated with a method.

Use this graph to measure and compare the extent of coverage provided by product or method across programs. The CYP factor is based on the percentage of one year where one unit of the method would provide a couple with contraception, if used properly.

You can compare coverage achievement over time between programs despite differences in method mix. A social marketing program may distribute more units and more volume and, perhaps, more value with methods like condoms and orals. But, a CYP analysis may indicate that a clinic NGO or MOH program may achieve more coverage because of the greater CYP factors of long-term methods such as Norplant, Depo-Provera, and IUDs.

Use this graph to evaluate and monitor relative program coverage or national coverage and achievement.

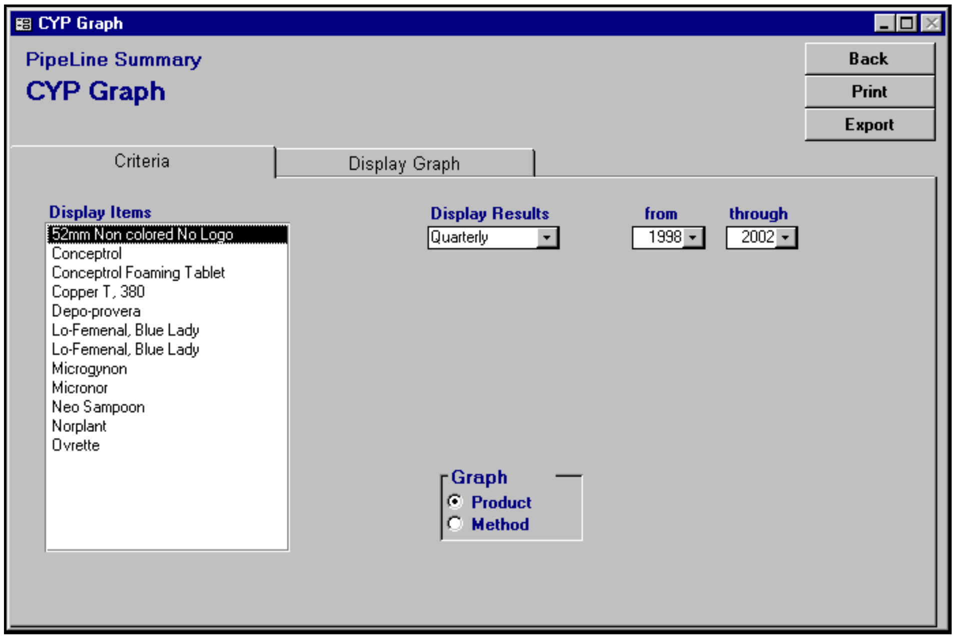
Creating the CYP Graph

Created the CYP graph is as follows.

From the PipeLine Summary Menu—

1. Click on the Couple-Years of Protection (CYP) Graph button.

Figure 65 - PipeLine Summary CYP graph screen


CYP Graph

PipeLine Summary
CYP Graph

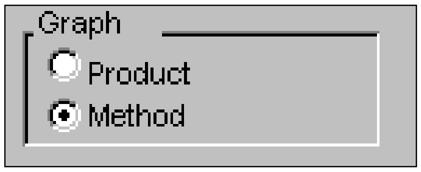
Display Items
52mm Non colored No Logo
Conceptrol
Conceptrol Foaming Tablet
Copper T, 380
Depo-provera
Lo-Femenal, Blue Lady
Lo-Femenal, Blue Lady
Microgynon
Micronor
Neo Sampoon
Norplant
Ovrette

Display Results: Quarterly
from: 1998
through: 2002

Graph
[✓ Product]
Method

The Display Items field shows the products or contraceptive methods that can be included in the graph.

Figure 66 - Graph type selection


Graph
Product
[✓ Method]

2. Click on the Method option to create a graph by method.

   Or click on the Product option to create a graph by product.

After you set the graph for products or methods—

3. Click on the product (or method) you want to include.

To select multiple products (or methods)—

4. Hold down the Control key (<Ctrl>) and click on each product or method you want to include.

Figure 68 — Export CYP Data window

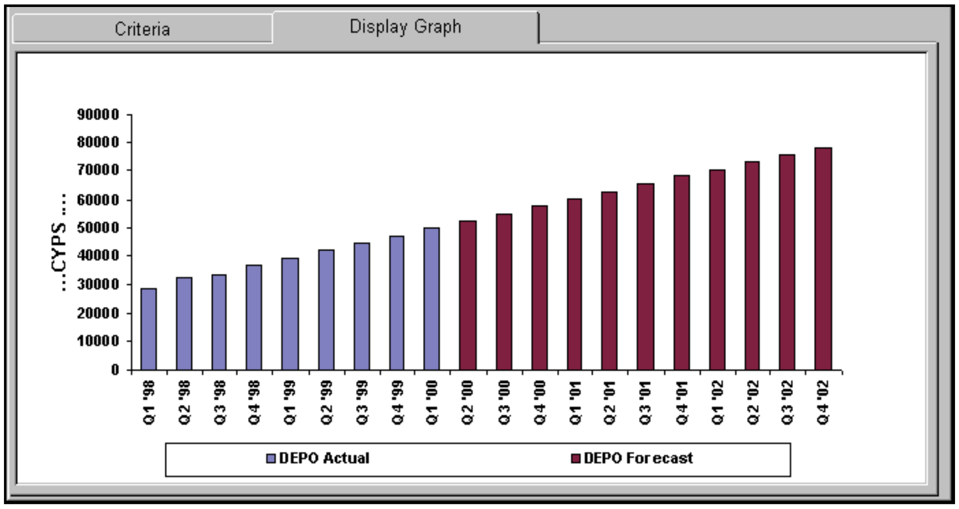
5. Click on the arrow next to the Display Results field, and select Monthly to display the data by month or Quarterly to display the data by quarter.
6. Click on the arrow next to the From field, and select the starting year of the report period from the pull-down menu.
7. Click on the arrow next to the Through field, and select the last year of the report period from the pull-down menu.

Displaying the Summary CYP Graph

Use the Display Graph tab to preview the CYP graph before printing it.

1. Click on the Display Graph tab.

Figure 67 - CYP Graph display



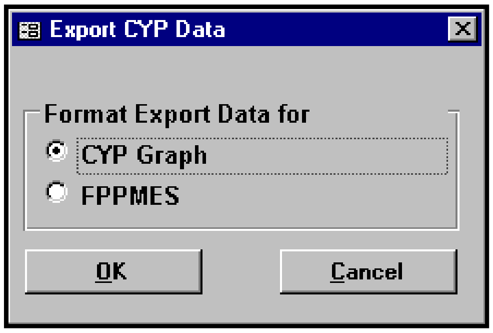
Exporting the CYP Graph

The Export button lets you export the data of a completed PipeLine Summary CYP graph. After you create a CYP graph, you can export the data as follows—

2. Click on the Export button.

PipeLine displays the Export CYP Data window. Use the options in this window to arrange the data in a Microsoft Office Excel spreadsheet.

Figure 69 - Export CYP Data Window



Selected: [✓ CYP Graph] FPPMES

CYP Graph
Incorporates the CYP factors into the consumption totals for ad hoc reports and graphs.

FPPMES
Arranges the data for the Family Planning Program Monitoring and Evaluation System.

The default is CYP Graph.

3. Click on the OK button to accept the default.

   Or, click on FPPMES, and click on the OK button.

PipeLine opens another Output window, so you can name and save the export file. The default export file name is shown in the File Name field.

4. Click on the OK button to accept the default.

   Or, click on the File Name field, and type the new export file name.

5. Click on the OK button to export, and save the Consumption graph.
6. Click on the Back button to return to the PipeLine Summary menu.

Previewing and Printing a Report or Graph
The procedure for printing a PipeLine Summary report or graph is the same regardless of the type you choose. To print a completed report or graph—

1. Click on the Print button to display the report or graph on your screen.
2. Click on the print icon to send the report or graph to your default printer.
3. Click on the close icon to return to the previous screen.
4. Click on the Back button to return to the PipeLine Summary menu.

APPENDIX C: THE REDDICK VBA (RVBA) NAMING CONVENTIONS, VERSION 6.01

The purpose of the Reddick VBA (RVBA) Naming Conventions is to provide a guideline for naming objects in the Visual Basic for Applications (VBA) language. Having conventions is valuable in any programming project. When you use them, the name of the object conveys information about the meaning of the object. These conventions attempt to provide a way of standardizing that meaning across the body of VBA programmers.

VBA is implemented to interact with a host application-for example, Microsoft Access, Microsoft Visual Basic, AutoCAD, and Visio. The RVBA conventions cover all implementations of the VBA language, regardless of the host application. Some of the tags described in this document may not necessarily have an implementation within some of the particular host programs for VBA. The word object, in the context of this document, refers to simple variables and VBA objects, as well as to objects made available by the VBA host program.

While I am the editor of these conventions, they are the work of many people, including Charles Simonyi, who invented the Hungarian conventions on which these are based, and Stan Leszynski, who co-authored several versions of the conventions. Many others, too numerous to mention, have also contributed to the development and distribution of these conventions, but I'd especially like to thank Paul Litwin and Ken Getz who have made substantial contributions over the years.

These conventions are intended as a guideline. If you disagree with a particular part of the conventions, simply replace that part with what you think works better. However, keep in mind that future generations of programmers may need to understand those changes, and place a comment in the header of a module indicating what changes have been made. To be concise, the conventions are presented without rationalizations for how they were derived although each of the ideas presented has a considerable history to it.

Changes to the Conventions

Some of the tags in the version of the conventions presented here have changed from previous versions. Consider all previous tags to be grandfathered into the conventions--you don't need to go back and make changes. For new development work, I leave it up to you to decide whether to use the older tags or the ones suggested here. In a few places in this document, older tags are shown in {braces}. As updates to this document are made, the current version can be found at the Xoc Software web site, http://www.xoc.net.

An Introduction to Hungarian

The RVBA conventions are based on the Hungarian conventions for constructing object names, named for the native country of the inventor, Charles Simonyi. The objective of Hungarian is to convey information about the object concisely and efficiently. Hungarian takes some getting used to, but once adopted, it quickly becomes second nature. The format of a Hungarian object name is

```
[prefixes]tag[BaseName[Suffixes]]
```

The square brackets indicate optional parts of the object name. These components have the following meanings:

Component	Meaning
Prefixes	Modify the tag to indicate additional information. Prefixes are all lowercase. They are usually picked from a standardized list of prefixes, given later in this document.
Tag	Short set of characters, usually mnemonic, that indicates the type of the object. The tag is all lowercase. It is usually selected from a standardized list of tags, given later in this document.
BaseName	One or more words that indicate what the object represents. Capitalize the first letter of each word in the BaseName.
Suffixes	Additional information about the meaning of the BaseName. Capitalize the first letter of each word in the Suffix. They are usually picked from a standardized list of suffixes, given later in this document.

Notice that the only required part of the object name is the tag. This may seem counterintuitive; you may feel that the BaseName is the most important part of the object name. However, consider a generic procedure that operates on any form. The fact that the routine operates on a form is the important thing, not what that form represents. Because the routine may operate on forms of many different types, you do not necessarily need a BaseName. However, if you have more than one object of a type referenced in the routine, you must have a BaseName on all but one of the object names to differentiate them. In addition, unless the routine is generic, the BaseName conveys information about the variable. In most cases, a variable should include a BaseName.

Tags

Use the techniques described in the following sections to construct tags to indicate the data type of an object.

Variable tags

Use the tags listed in Table 21 for VBA data types. You can also use a specific tag instead of *obj* for any data type defined by the host application or one of its objects. (See the section "Host Application and Component Extensions to the Conventions" later in this document.)

Table 21 - Tables for VBA Variables

Tag	Object Type
bool {f, bln}	Boolean
byte {byt}	Byte
cur	Currency
date {dtm}	Date
dec	Decimal
dbl	Double
int	Integer
lng	Long
obj	Object
sng	Single
str	String
stf	String (fixed length)
var	Variant

Here are several examples:

```
lngCount
intValue
strInput
```

You should explicitly declare all variables, each on a line by itself. Do not use the old-type declaration characters, such as %, &, and $. They are extraneous if you use the naming conventions, and there is no character for some of the data types, such as Boolean. You should always explicitly declare all variables of type Variant using the *As Variant* clause, even though it is the default in VBA. For example:

```
Dim intTotal As Integer
Dim varField As Variant
Dim strName As String
```

Constructing Properties Names

Properties of a class present a particular problem: should they include the naming convention to indicate the type? To be consistent with the rest of these naming conventions, they should. However, it is permitted to have property names without the tags, especially if the class is to be made available to customers who may not be familiar with these naming conventions.

Collection Tags

You treat a collection object with a special tag. You construct the tag using the data type of the collection followed by the letter s. For example, if you had a collection of Longs, the tag is lngs. If it was a collection of forms, the tag for the collection is frms. Although, in theory, a collection can hold objects of different data types, in practice, each of the data types in the collection is the same. If you do want to use different data types in a collection, use the objs tag. For example:

```
intsEntries
frmsCustomerData
objsMisc
```

Constants

Constants always have a data type in VBA. Because VBA will choose this data type for you if you don't specify it, you should always specify the data type for a constant. Constants declared in the General Declarations section of a module should always have a scope keyword of Private or Public, and be prefixed by the scope prefixes *m* or *g*, respectively. A constant is indicated by appending the letter *c* to the end of the data type for the constant. For example:

```
Const intcGray As Integer = 3
Private Const mdblcPi As Double = 3.14159265358979
```

Although this technique is the recommended method of naming constants, if you are more concerned about specifying that you are dealing with constants rather than their data type, you can alternatively use the generic tag *con* instead. For example:

```
Const conPi As Double = 3.14159265358979
```

Menu Items

The names of menu items should reflect their position in the menu hierarchy. All menu items should use the tag mnu, but the BaseName should indicate where in the hierarchy the menu item falls. Use *Sep* in the BaseName to indicate a menu separator bar, followed by an ordinal. For example:

```
mnuFile (on menu bar)
mnuFileNew (on File popup menu)
mnuFileNewForm (on File New flyout menu)
mnuFileNewReport (on File New flyout menu)
mnuFileSep1 (first separator bar on file popup menu)
mnuFileSaveAs (on File popup menu)
mnuFileSep2 (second separator bar on file popup menu)
mnuFileExit (on File popup menu)
mnuEdit (on menu bar)
```

Creating Data Types

VBA gives you three ways to create new data types: enumerated types, classes, and user-defined types. In each case, you will need to invent a new tag that represents the data type that you create.

Enumerated types

Groups of constants of the *long* data type should be made an enumerated type. Invent a tag for the type, append a "c," and then define the enumerated constants using that tag. Because the name used in the Enum line is seen in the object browser, you can add a BaseName to the tag to spell out the abbreviation indicated by the tag. For example:

```
Public Enum ervcErrorValue
ervcInvalidType = 205
ervcValueOutOfBounds
End Enum
```

The BaseName should be singular, so that the enumerated type should be ervcErrorValue, not ervcErrorValues. The tag that you invent for enumerated types can then be used for variables that can contain values of that type. For example:

```
Dim erv As ervcErrorValue
Private Sub Example(ByVal ervCur As ervcErrorValue)
```

While VBA only provides enumerated types of groups of the long type, you can still create groups of constants of other types. Just create a set of constant definitions using an invented tag. For example:

```
Public Const estcError205 As String = "Invalid type"
Public Const estcError206 As String = "Value out of bounds"
```

Unfortunately, because this technique doesn't actually create a new type, you don't get the benefit of the VBA compiler performing type checking for you. You create variables that will hold constants using a similar syntax to variables meant to hold instances of enumerated types. For example:

```
Dim estError As String
```

Tags for classes and user-defined types

A class defines a user-defined object. Because these invent a new data type, you will need to invent a new tag for the object. You can add a BaseName to the tag to spell out the abbreviation indicated by the tag. User-defined types are considered a simple class with only properties, but in all other ways are used the same as class modules. For example:

```
gphGlyph
edtEdit
Public Type grbGrabber
```

You then define variables to refer to instances of the class using the same tag: For example:

```
Dim gphNext As New gphGlyph
Dim edtCurrent as edtEdit
Dim grbHandle as grbGrabber
```

Polymorphism

In VBA, you use the Implements statement to derive classes from a base class. The tag for the derived class should use the same tag as the base class. The derived classes, though, should use a different BaseName from the base class. For example:

```
anmAnimal (base class)
anmZebra (derived class of anmAnimal)
anmElephant (derived class of anmAnimal)
```

This logic of naming derived classes is used with forms, which are all derived from the pre-defined Form base class and use the frm tag. If a variable is defined to be of the type of the base class, then use the tag, as usual. For example:

```
Dim anmArbitrary As anmAnimal
Dim frmNew As Form
```

On the other hand, if you define a variable as an instance of a derived class, include the complete derived class name in the variable name. For example:

```
Dim anmZebraInstance As anmZebra
Dim anmElephantExample As anmElephant
Dim frmCustomerData As frmCustomer
```

Constructing Procedures

VBA procedures require you to name various items: procedure names, parameters, and labels. These objects are described in the following sections.

Constructing Procedure Names

VBA names event procedures, and you cannot change them. You should use the capitalization defined by the system. For user-defined procedure names, capitalize the first letter of each word in the name. For example:

```
cmdOK_Click
GetTitleBarString
PerformInitialization
```

Procedures should always have a scope keyword, Public or Private, when they are declared. For example:

```
Public Function GetTitleBarString() As String
Private Sub PerformInitialization
```

Naming Parameters

You should prefix all parameters in a procedure definition with ByVal or ByRef, even though ByRef is optional and redundant. Procedure parameters are named the same as simple variables of the same type, except that arguments passed by reference use the prefix "r." For example:

```
Public Sub TestValue(ByVal intInput As Integer, ByRef rlngOutput As Long)
Private Function GetReturnValue(ByVal strKey As String, _
ByRef rgph As Glyph) As Boolean
```

Naming Labels

Labels are named using upper and lower case, capitalizing the first letter of each word. For example:

```
ErrorHandler:
ExitProcedure:
```

Prefixes

Prefixes modify an object tag to indicate more information about an object.

Arrays of Objects Prefix

Arrays of an object type use the prefix "a." For example:

```
aintFontSizes
astrNames
```

Index Prefix

You indicate an index into an array by the prefix i, and for consistency the data type should always be a long. You may also use the index prefix to index into other enumerated objects, such as a collection of user-defined classes. For example:

```
iaintFontSizes
iastrNames
igphsGlyphCollection
```

Prefixes for Scope and Lifetime

Three levels of scope exist for each variable in VBA: Public, Private, and Local. A variable also has a lifetime of the current procedure or the lifetime of the object in which it is defined. Use the prefixes in Table 22 to indicate scope and lifetime.

Table 22 - Scope prefixes

Prefix	Object Type
(none)	Local variable, procedure-level lifetime, declared with “Dim”
s	Local variable, object lifetime, declared with “Static”
m	Private (module) variable, object lifetime, declared with “Private”
g	Public (global) variable, object lifetime, declared with “Public”

You also use the “m” and “g” constants with other objects, such as constants, to indicate their scope. For example:

```
intLocalVariable
mintPrivateVariable
gintPublicVariable
mdblcPi
```

VBA allows several type declaration words for backward compatibility. The older keyword “Global” should always be replaced by “Public,” and the “Dim” keyword in the General Declarations section should be replaced by “Private.”

Other Prefixes

Table 23 lists and describes some other prefixes:

Table 23 - Other commonly-used prefixes

Prefix	Object Type
c	Count of some object type
h	Handle to a Windows object
r	Parameter passed by reference

Here are some examples:

```
castrArray
hWndForm
```

Suffixes

Suffixes modify the base name of an object, indicating additional information about a variable. You’ll likely create your own suffixes that are specific to your development work. Table 24 lists some generic VBA suffixes.

Table 24 - Commonly-used suffixes

Suffix	Object Type
Min	The absolute first element in an array or other kind of list
First	The first element to be used in an array or list during the current operation
Last	The last element to be used in an array or list during the current operation
Lim	The upper limit of elements to be used in an array or list. Lim is not a valid index.

	Generally, Lim equals Last + 1
Max	The absolutely last element in an array or other kind of list
Cnt	Used with database elements to indicate that the item is a Counter. Counter fields are incremented by the system and are numbers of either type Long or type Replication Id.

Here are some examples:

```
iastrNamesMin iastrNamesMax
iaintFontSizesFirst
igphsGlyphCollectionLast
lngCustomerIdCnt varOrderIdCnt
```

File Names

When naming items stored on the disk, no tag is needed because the extension already gives the object type. For example:

```
Test.Frm (frmTest form)
Globals.Bas (globals module)
Glyph.Cls (gphGlyph class module)
```

Host Application and Component Extensions to the Conventions

Each host application for VBA, as well as each component that can be installed, has a set of objects it can use. This section defines tags for the objects in the various host applications and components.

Access 2000, Version 9.0 Objects

Table 25 lists Access object variable tags. Besides being used in code to refer to these object types, these same tags are used to name these kinds of objects in the form and report designers.

Table 25 - Access object variable tags

Tag	Object Type
aob	AccessObject
aops	AccessObjectProperties
aop	AccessObjectProperty
app	Application
bfr	BoundObjectFrame
chk	CheckBox
cbo	ComboBox
cmd	CommandButton
ctl	Control
ctls	Controls
ocx	CustomControl
dap	DataAccessPage
dcm	DoCmd
frm	Form
fcd	FormatCondition
fcds	FormatConditions
frms	Forms
grl	GroupLevel
hyp	Hyperlink
img	Image
lbl	Label
lin	Line
lst	ListBox
bas	Module
ole	ObjectFrame
opt	OptionButton
fra	OptionGroup (frame)
brk	PageBreak
pal	PaletteButton
prps	Properties
shp	Rectangle
ref	Reference
refs	References
rpt	Report
rpts	Reports
scr	Screen
sec	Section
sfr	SubForm

Tag	Object Type
srp	SubReport
tab	TabControl
txt	TextBox
tgl	ToggleButton

Some examples:

```
txtName
lblInput
```

For ActiveX custom controls, you can use the tag ocx as specified in Table 25 or more specific object tags that are listed later in this document in Tables 34 and 35. For an ActiveX control that doesn't appear in the Tables 34 or 35, you can either use ocx or invent a new tag.

DAO 3.6 Objects

DAO is the programmatic interface to the Jet database engine shared by Access, Visual Basic, and Visual C++. The tags for DAO 3.6 objects are shown in Table 26.

Table 26 - DAO object tags

Tag	Object Type
cnt	Container
cnts	Containers
db	Database
dbs	Databases
dbe	DBEngine
doc	Document
docs	Documents
err	Error
errs	Errors
fld	Field
flds	Fields
grp	Group
grps	Groups
idx	Index
idxs	Indexes
prm	Parameter
prms	Parameters
pdbe	PrivDBEngine
prp	Property
prps	Properties
qry	QueryDef
qrys	QueryDefs

Tag	Object Type
rst	Recordset
rsts	Recordsets
rel	Relation
rels	Relations
tbl	TableDef
tbls	TableDefs
usr	User
usrs	Users
wrk	Workspace
wrks	Workspaces

Here are some examples:

```
rstCustomers
idxPrimaryKey
```

Table 27 lists the tags used to identify types of objects in a database.

Table 27 - Access Database Explorer object tags

Tag	Object Type
tbl	Table
qry	Query
frm	Form
rpt	Report
mcr	Macro
bas	Module
dap	DataAccessPage

If you wish, you can use tags that are more exact or suffixes to identify the purpose and type of a database object. If you use the suffix, use the tag given from Table 27 to indicate the type. Use either the tag or the suffix found along with the more general tag, but not both. The tags and suffixes are shown in Table 28.

Table 28 - Specific object tags and suffixes for Access Database Explorer objects

Tag	Suffix	Object Type
tlkp	Lookup	Table (lookup)
qsel	(none)	Query (select)
qapp	Append	Query (append)
qxtb	XTab	Query (crosstab)
qddl	DDL	Query (DDL)
qdel	Delete	Query (delete)
qflt	Filter	Query (filter)
qlkp	Lookup	Query (lookup)
qmak	MakeTable	Query (make table)

Tag	Suffix	Object Type
qspt	PassThru	Query (SQL pass-through)
qtot	Totals	Query (totals)
quni	Union	Query (union)
qupd	Update	Query (update)
fdlg	Dlg	Form (dialog)
fmnu	Mnu	Form (menu)
fmsg	Msg	Form (message)
fsfr	SubForm	Form (subform)
rsrp	SubReport	Form (subreport)
mmnu	Mnu	Macro (menu)

Here are some examples:

```
tblValidNamesLookup
tlkpValidNames
fmsgError mmnuFileMnu
```

When naming objects in a database, do not use spaces. Instead, capitalize the first letter of each word. For example, instead of Quarterly Sales Values Table, use tblQuarterlySalesValues.

There is strong debate over whether fields in a table should have tags. Whether you use them is up to you. However, if you do use them, use the tags from Table 29.

Table 29 - Field tags (if you decide to use them)

Tag	Object Type
lng	Autoincrementing (either sequential or random) Long (used with the suffix Cnt)
bin	Binary
byte	Byte
cur	Currency
date	Date/time
dbl	Double
guid	Globally unique identified (GUID) used for replication AutoIncrement fields
int	Integer
lng	Long
mem	Memo
ole	OLE
sng	Single
str	Text
bool	Yes/No

Visual Basic 6.0 Objects

Table 30 shows the tags for Visual Basic 6.0 objects.

Table 30 - Visual Basic 6.0 object tags

Tag	Object Type
app	App
chk	CheckBox
clp	Clipboard
cbo	ComboBox
cmd	CommandButton
ctl	Control
dat	Data
dir	DirListBox
drv	DriveListBox
fil	FileListBox
frm	Form
fra	Frame
glb	Global
hsb	HScrollBar
img	Image
lbl	Label
lics	Licenses
lin	Line
lst	ListBox
mdi	MDIForm
mnu	Menu
ole	OLE
opt	OptionButton
pic	PictureBox
prt	Printer
prp	PropertyPage
scr	Screen
shp	Shape
txt	TextBox
tmr	Timer
uctl	UserControl
udoc	UserDocument
vsb	VscrollBar

Microsoft ActiveX Data Objects 2.1 Tags

Office 2000 provides version 2.1 of the ActiveX Data Objects library. Table 31 lists the recommended tags for this version of ADO.

Note: Many of the ADO, ADOX, and JRO tags overlap with existing DAO tags. Make sure you include the object library name in all references in your code, so there's never any possibility of confusion. For example, use

```
 Dim rst As ADODB.Recordset
```

or

```
 Dim cat As ADOX.Catalog
```

rather than using the object types without the library name. This will not only make your code more explicit and avoid confusion about the source of the object, but will also make your code run a bit faster.

Table 31 - ADO 2.1 Object tags

Tag	Object Type
cmn {cmd}	Command
cnn {cnx}	Connection
err	Error
errs	Errors
fld	Field
flds	Fields
prm	Parameter
prms	Parameters
prps	Properties
prp	Property
rst	Recordset

Microsoft ADO Ext. 2.1 for DDL and Security (ADOX) Tags

In order to support DDL and security objects within Jet database, Microsoft provides ADOX, an additional

ADO library of objects. Table 32 lists tags for the ADOX objects.

Table 32 - ADOX Object tags

Tag	Object Type
cat	Catalog
clms	Column
clm	Columns
cmd	Command
grp	Group
grps	Groups
idx	Index
idxs	Indexes
key	Key
keys	Keys
prc	Procedure
prcs	Procedures
prps	Properties
prp	Property
tbl	Table
tbls	Tables
usr	User
usrs	Users
vw	View
vws	Views

Microsoft Jet and Replication Objects 2.1

In order to support Jet's replication features, ADO provides another library (JRO). Table 33 lists suggested tags for the JRO objects.

Table 33 - JRO object tags

Tag	Object Type
flt	Filter
flts	Filters
jet	JetEngine
rpl	Replica

Microsoft SQL Server and Microsoft Data Engine (MSDE) Objects

Table 34 lists RVBA tags for Microsoft SQL Server and the Microsoft Data Engine (a limited-connection version of SQL Server 7) objects.

Table 34 - SQL Server/MSDE object tags

Tag	Object Type
tbl	table
proc	stored procedure
trg	trigger
qry	view
dgm	database diagram
pk	primary key
fk	foreign key
idx	other (non-key) index
rul	check constraint
def	default

Microsoft Common Control Objects

Windows 95 and Windows NT have a set of common controls that are accessible from VBA. Table 35 lists the tags for objects created using these controls.

Table 35 - Microsoft Common Control Object tags

Tag	Object Type
ani	Animation
btn	Button (Toolbar)
bmn	ButtonMenu (Toolbar)
bmns	ButtonMenus (Toolbar)
bnd	Band (CoolBar)
bnds	Bands (CoolBar)
bnp	BandsPage (CoolBar)
btns	Buttons (Toolbar)
cbr	CoolBar
cbp	CoolBarPage (CoolBar)
hdr	ColumnHeader (ListView)
hdrs	ColumnHeaders (ListView)
cbi	ComboItem (ImageCombo)
cbis	ComboItems (ImageCombo)
ctls	Controls
dto	DataObject
dtf	DataObjectFiles
dtp	DTPicker
fsb	FlatScrollBar
imc	ImageCombo
iml	ImageList

Tag	Object Type
lim	ListImage
lims	ListImages
lit	ListItem (ListView)
lits	ListItems (ListView)
lsi	ListSubItem (ListView)
lsis	ListSubItems (ListView)
lvw	ListView
mvw	MonthView
nod	Node (TreeView)
nods	Nodes (TreeView)
pnl	Panel (Status Bar)
pnls	Panels (Status Bar)
prb	ProgressBar
sld	Slider
sbr	StatusBar
tab	Tab (Tab Strip)
tabs	Tabs (Tab Strip)
tbs	TabStrip
tbr	Toolbar
tvw	TreeView
udn	UpDown

Other Custom Controls and Objects

Finally, Table 36 lists the tags for other commonly used custom controls and objects.

Table 36 - Tags for commonly-used custom controls

Tag	Object Type
cdl	CommonDialog (Common Dialog)
dbc	DBCombo (Data Bound Combo Box)
dbg	DBGrid (Data Bound Grid)
dls	DBList (Data Bound List Box)
gau	Gauge (Gauge)
gph	Graph (Graph)
grd	Grid (Grid)
msg	MAPIMessages (Messaging API Message Control)
ses	MAPISession (Messaging API Session Control)
msk	MaskEdBox (Masked Edit Textbox)
key	MhState (Key State)
mmc	MMControl (Multimedia Control)

Tag	Object Type
com	MSComm (Communication Port)
out	Outline (Outline Control)
pcl	PictureClip (Picture Clip Control)
rtf	RichTextBox (Rich Textbox)
spn	SpinButton (Spin Button)

Summary

Using a naming convention requires a considerable initial effort on your part. The payoff comes when either you or another programmer has to revisit your code later. Using the conventions given here will make your code more readable and maintainable.

Greg Reddick is the President of Xoc Software, a software development company developing programs in Visual Basic, Microsoft Access, C/C++, and for the web. He leads training seminars in Visual Basic for Application Developers Training Company (AppDev). In a previous life, he worked for four years on the Access development team at Microsoft. Greg can be reached at mailto:grr@xoc.net or from the Xoc Software web site, http://www.xoc.net.

APPENDIX D: THE REDDICK VBA (RVBA) CODING CONVENTIONS (VERSION 0.90)

What follows are the Reddick VBA (RVBA) Coding Conventions. The objectives of the conventions are to make code:

More readable: Conventions allow a reader to understand the meaning of the code with less effort.

- More maintainable: The code can be more reliably changed to fix bugs and enhance functionality.
- More reliable: The code is more likely to perform as expected.
- More efficient: The code performs faster or consumes fewer resources.

These conventions are separate from the RVBA Naming Conventions and may be adopted without adopting the naming conventions. The current version of these conventions can always be found on the Xoc Software web site: http://www.xoc.net.

More rational is provided for these recommendations than is given in the RVBA Naming Conventions. In most cases, there are good rationales for the given conventions. However, in some cases an arbitrary decision was made to select one convention from a set of reasonable alternatives. The other reasonable alternatives to the conventions placed in {braces} at the end of a section. In some cases, a topic only relates to the Visual Basic 6.0 development environment, as opposed to VBA in general. In those cases, the topic is marked with [VB6] after the topic heading.

No set of conventions can cover every case or every consideration. The general rule is that exceptions to the conventions can be made with the approval of the programming team after careful consideration.

The sections are listed in alphabetical order to facilitate their use as a reference work. However, this makes the flow of the document unusual for casual reading as some topics are much more technical than others.

Arrays

Always specify the both the lower and upper bound of an array. This makes explicit whether element zero of the array is a valid element or not. For example:

```
Dim astrValue(1 To 10) As String
```

By convention the index variable used to walk an array should always be a Long data type. This assures that if the array size grows past 32767 elements when maintaining the program that the index variable can still address all elements in the array

When walking an array, always use the VBA LBound and UBound functions to visit each item. This makes sure that every item in the array is visited. For example:

```
Dim iastrValue As Long
For iastrValue = LBound(astrValue) To UBound(astrValue)
```

```
MsgBox astrValue(iastrValue)
Next iastrValue
```

Assertions

VBA provides a built-in assertion mechanism through Debug.Assert. If the expression following the Debug.Assert evaluates to True, the code continues. If the expression evaluates to False, VBA enters Break mode as if a breakpoint had been set on that line. The line shown here acts as a hard coded breakpoint:

```
Debug.Assert False
```

Assertions that do not include a function call in the expression are removed by the compiler when an executable is made, so they only apply to debugging inside the VBA environment. Assertions with a function call in the expression will remain in the executable, but the resulting value of the expression is discarded. VBA doesn't remove function calls because they may have side effects, but discards the return value from the function.

Any time that there is an assumption in the code about the state of the program, there should be an assertion that states the assumption. For example, suppose that a procedure includes this code:

```
Select Case intValue
Case 1
MsgBox "Aircraft"
Case 2
MsgBox "AutoMobile"
Case 3
MsgBox "SnowMobile"
End Select
```

This code assumes that the value of intValue is between one and three. However, if through some bug, intValue had the value of zero or four, this code doesn't work right. The result is that no MsgBox appears at all. Tracking down why the MsgBox doesn't appear is time consuming. Instead, the code could be written one of two other ways. Either:

```
Debug.Assert intValue >= 1 And intValue <= 3
Select Case intValue
Case 1
MsgBox "Aircraft"
Case 2
MsgBox "AutoMobile"
Case 3
MsgBox "SnowMobile"
End Select
```

Or

```
 Select Case intValue
 Case 1
 MsgBox "Aircraft"
 Case 2
 MsgBox "AutoMobile"
 Case 3
 MsgBox "SnowMobile"
 Case Else
 Debug.Assert False
 End Select
```

In general, every Select/End Select block should have a Case Else to trap unexpected values. If the Case Else should never occur, then a Debug.Assert False should be inserted into the block. If the code is correctly written to handle 1 To 3, but zero and four are allowed values, the code should be written with a comment in the Case Else block to indicate that this is expected, like this:

```
 Select Case intValue
 Case 1
 MsgBox "Aircraft"
 Case 2
 MsgBox "AutoMobile"
 Case 3
 MsgBox "SnowMobile"
Case Else 'Do nothing
End Select
```

Assertions trap logic errors early. Rather than waiting to see the results of a bug in the use interface, there is immediate feedback that the bug has occurred. Assertions are only effective if they are present, which means that they have to be added when writing the code. Any logic error that is fixed in the code is a good indication that some additional assertions need to be added.

Comments

A comment in VBA starts with an apostrophe and ends at the end of the line. Comments may be placed on a line by themselves or at the end of a line. A comment starts with the apostrophe followed immediately by the text with no space between the two.

The comment at the end of a line should be used in only a few places:

- At the end of a declaration line
- On a Case line
- On the line that ends a block to indicate what block is being ended. For example on a set of nested If/End If blocks, a comment on the End If line may say what If block is completed. This is especially useful if the block spans several screens.

Examples:

```
 Dim dateUTC As String 'time in Univeral Coordinated Time Case 11 'Division by Zero
```

If the end of a line comment line exceeds the 80 characters line limit, continue the comment on the next line indented by one tab stop. For example:

```
 Case 35602 'This key is already associated with an element of this 'collection Set
nodChild = tvw.Nodes.Item(cci.Guid)
```

All other comments should be placed on a separate line above the line they are documenting and indented to the same level. A comment of this sort is generally preceded by a blank line unless it is the first line of an indented block. For example:

```
 vt = vti.VarType

 'Special hack for analyzing my code If LCase$(Left$(strParamName,
Len(strcDecPrefix))) = strcDecPrefix Then strDataType = strDecimal End If
```

If it is the first line of an indented block, it is not preceded by a blank line. For example:

```
 If mboolShowProperties Then 'Show properties for each member For Each mi In
ci.Members
```

Comments should state the intention of the code not how it performs the task. This is an example of a worthless comment:

```
 'Place the VarType into the vt variable vt = vti.VarType
```

It is worse than no comment at all. The comment is wrong if the code changes to use the variable name vtCur instead of vt without changing the comment. When reading a comment that doesn't match the code, the question becomes whether the comment is correct or the code is correct. Usually it is the comment that is wrong, but it may take some time to prove that. A wrong comment can be worse than no comment at all. A comment that says the same thing as the following line of code is worthless. In general, don't write comments that have to be maintained, because in the real-world comments frequently aren't maintained.

A comment that states the intention of the code, though, may be useful. For example:

```
 'Store VarType for recovery in error condition. vt = vti.VarType
```

However, use these comments only when the intention is not immediately clear when reading the code. Instead strive to make the code self explanatory, through good naming and coding conventions.

Constants

Always give constants an explicit data type. For example:

```
Private Const dblcPi As Double = 3.14159265358979
```

If a literal value other than zero or one appears in the code, consideration should be given to whether it makes things more readable and maintainable to replace it with a constant. Replace a magic number used more than once in the code with a constant.

Global constants are allowed and encouraged. Replace sets of constants of the data type long with enumerated types using Enum.

Date Functions and Date Variables

Be careful about using the VBA date functions: Date, DateAdd, DateSerial, DateValue, and Now. These functions return a variant containing a date. If implicit type conversion to turns the return value into a string, the string representation of the date displays a two-digit year number. That year number is, of course, not Y2K compliant. This also applies to allowing a variable of type Date to be converted into a string. Instead, use the Format$ function to convert the date into a string. For example:

```
strValue = Format$(Date, "mm/dd/yyyy")
```

Default Properties

Using default properties makes code difficult to read. VBA allows you to just use the name of a textbox and looks up the default property, Text. For example:

```
MsgBox txtValue
```

This prints the value of the txtValue textbox. On the other hand, it is much clearer to say:

```
MsgBox txtValue.Text
```

To even be more explicit, it could even be expressed as:

```
MsgBox Me.txtValue.Text
```

This, however, does not add any additional worth because all references to a control in a module from a form are implicitly on Me.

The reason to be explicit about default properties is to keep the programmer from having to figure out what property is being referenced. This is especially true when referencing ActiveX controls and ActiveX DLLs where the default properties are obscure. For example, when an ADO field is referenced, you are allowed to say:

```
varValue = rst!strFirstName
```

This references the Value property of the strFirstName field. However, it is much clearer to say:

```
varValue = rst.Fields.Item("strFirstName").Value
```

This code doesn't use any default properties and retrieves the same value.

Deprecated Features

Avoid using features Visual Basic supports only for backwards compatibility. Avoid using undocumented features. Also, avoid using functionality that VBA has replaced with functionality that is more modern. Some examples of these kinds of features:

- %, &, $, Etc. in declaration of variables
- Rem statements
- Line numbers (except in conjunction with the Erl function in special error handling situations)
- Single line If statements (use If/End If blocks instead). For example, don't use:

```
If boolValue Then MsgBox "Hi There"
```

- While/Wend loops (replace with Do While/Loop)

- Variables declared with Global (use Public instead). Using Dim in the General Declarations section (use Private instead)
- Using user defined types except in the case of Windows API calls or reading fixed width record files (use Class modules instead)
- Gosub
- The End statement in most cases (simply unload the last form in a standard EXE instead)

Disambiguation

When referencing classes from an ActiveX library, always use the library name to explicitly tell VBA from what library to get the class. If you don't, then VBA will use the order of the libraries in the References dialog to determine from which library it gets the class. The library name always appears in the upper left-hand listbox of the VBA object browser. For example, if there are references to both the Access and Excel object libraries, then this is ambiguous:

```
Dim appObj As Application
```

Because both the Access and Excel libraries include an Application class, which Application class is referenced depends on which one appears first in the References dialog. Instead, it should be declared like this:

```
Dim appObj As Excel.Application
```

Microsoft refers to this as "disambiguation". With this declaration, it does not matter what the order of the libraries is inside the References dialog, as appObj will always refer to the Excel Application object. All references to class names in libraries should include the disambiguating library name.

DLL Base Address [VB6]

The base address is the location that the DLL is loaded into memory. If two DLLs are loaded into the same base address, then VBA moves the second DLL to a new address. VBA then has to modify the binary code within the DLL's address space to reflect the new address. This slows down loading the second DLL.

Libraries used together should start at different base addresses. In Visual Basic, enter the Base Address for a library in the Project Properties dialog Compile tab. Enter a random number base address different than any other used at the same time.

Dollar Sign ($) Functions

If the result of a function is used as a string or assigned it to a string variable, use the $ form of the function. This results in faster executing code, because a conversion from a variant to a string is unnecessary. For example, this is proper usage of dollar sign functions:

```
 If LCase$(Left$(strParamName, Len(strcDecPrefix))) = strcDecPrefix Then
```

This example calls the LCase$ and Left$ functions instead of the LCase and Left functions because the result is used as a String. If the result is used as a Variant, then call the LCase and Left functions instead.

The $ version of the function returns the same value as the Variant version. The one except to the rule is the VBA Date function. The Date function should always be used because the Date$ function doesn't behave correctly. The Date$ always returns information in mm-dd-yyyy format regardless of the Windows localization settings, whereas the Date function uses the localization settings.

Error Handling

A procedure should always include runtime error handling. In general, Error handling should be blocked out the same way in every procedure, as shown in this example:

```
Private Sub Test() On Error GoTo ErrorHandler
 'Code for the procedure goes here
ExitProcedure:
 On Error Resume Next
 'Cleanup code for the procedure goes here Exit Sub ErrorHandler:
 Select Case Err.Number
 'Case statements for expected errors goes here
 Case Else
 Call UnexpectedError(Err.Number, Err.Description, Err.Source, _
 Err.HelpFile, Err.HelpContext)
 End Select
 Resume ExitProcedure End Sub
```

Use the label names shown in the example, although the label names have been arbitrarily chosen. Notice that the Exit Sub and ErrorHandler label are left justified making them easily findable. Case statements for expected errors should be given with the error number and a comment with the error message. For example:

```
 Select Case Err.Number
 'Case statements for expected errors go here
 Case 11 'Division by zero
 MsgBox "Zero isn't a valid divisor", vbExclamation, Me.Caption
 Case Else
```

The UnexpectedError routine is a global routine that is only called in a condition where a runtime error that isn't expected is received, so that there is a bug in the problem. This procedure should log the error message. At the absolute minimum it should just look like this, but ideally it should do a lot more to log the error:

```
Public Sub UnexpectedError(ByVal lngNumber As Long, _

 ByVal strDescription As String, ByVal strSource As String, _ ByVal strHelpfile As
String, ByVal lngHelpContext As Long) On Error Resume Next MsgBox "[" & strSource &
"]" & vbCrLf & "Run-time error '" _

 & CStr(lngNumber) & "':" _ & vbCrLf & vbCrLf & strDescription, vbExclamation,
App.Title, _ strHelpfile, lngHelpContext

 Debug.Print "Case " & CStr(lngNumber) & " '" & strDescription Debug.Assert False End
Sub
```

The first executable line of every procedure should be the On Error GoTo ErrorHandler line. The only exception to the rule is when a procedure checks the values of its arguments and generates a runtime error when they are invalid. In this case, the checking code comes before the On Error GoTo line. For example

```
Public Sub Test(ByVal intValue As Integer) If intValue < 1 Or intValue > 10 Then Call
Err.Raise(Number:=lngcInvalidValue, _

 Description:=strcInvalidValue) End If On Error Goto ErrorHandler
```

Exiting a Procedure

In general, a procedure should only have one exit point. Having one exit point makes it easier to read the code and understand when and where it exits. If you use the code mentioned in the Error Handling code section, that exit point is the Exit Sub, Exit Function, or Exit Property line at the top of the error handling. The only other way to exit the procedure should be through using Err.Raise. These Err.Raise lines should occur either before the On Error GoTo line when validating the parameters (see Error Handling) or inside the error handler.

In a few cases, there may be a need to raise an error inside the body of the procedure. In such cases, you should explicitly set any object variables to Nothing (see Nothing), and then exit the procedure. In such cases, the exiting the procedure should be explicitly detailed by a comment that shows the exit, consisting of an arrow stretching to 80 character right margin. The On Error GoTo 0 statement has to be used to turn off error handling for this procedure before executing the Err.Raise. For example, if this code appears somewhere after the On Error GoTo line, it should be written like this to make it explicit that there is an exit point in the middle of the procedure:

```
 If intValue > 1000 Then 'Raise an Error--------------------------------------------
---------> Set rst = Nothing On Error Goto 0 Call Err.Raise(Number:=lngcInvalidValue,
_

 Description:=strcInvalidValue) End if
```

For/Next and For Each/Next Loops

The index variable used in the For/Next loop should be specified on the Next line. This makes it explicit which For loop is being completed. For example:

```
 Dim iastrValue As Long For iastrValue = LBound(astrValue) To UBound(astrValue) MsgBox
astrValue(iastrValue) Next iastrValue
```

The object variable used to walk the collection should be placed on the Next line in a For Each/Next loop. For example:

```
 Dim frm As Form For Each frm in Forms

 If Not (frm Is Me) Then Unload Me End If Next frm
```

GoTo Statements

You can usually avoid using GoTo statements in VBA code. Use GoTo statements only when the alternative code is not as clear as the GoTo statement. A common reason to use a GoTo statement is to jump out of nested loops. For Example:

```
 For iastrOuterLoop = 1 To 10

 For iastrInnerLoop = 1 To 100 'some other code If astr(iastrOuterLoop,
iastrInnerLoop) = "Done" Then

 GoTo ExitNestedLoops End If 'some other code

 Next iastrInnerLoop Next iastrOuterLoop ExitNestedLoops: 'More code here
```

Headers

Each module should start with a header code that looks something like this:

```
'$Header: $
'****************************************************************************** Option
Explicit 'This module includes definitions of Windows API calls
```

The line of asterisks is an apostrophe followed by 79 asterisks. See the section on Long Lines.

Each Public procedure should begin with a header block that looks something like this:

```
Public Sub Almanac(ByVal lngTrecena As Long, ByVal vein As veinc, _ ByVal lngRows As
Long, ByRef alngBlack() As Long, _ ByRef alngRed() As Long, ByRef aveinRowStart() As
veinc, _ ByRef aveinAlmanac() As veinc, ByRef lngComplete As Long) 'Generates a Maya
almanac
```

'If lngComplete returns zero then it is an almanac, if it is non-zero, 'then it misses completing and you'll need to report that. You will still 'need to handle the error encNotAnAlmanac because the black numbers 'in alngBlack must wrap back to the starting lngTrecena. On Error GoTo ErrorHandler

Read/write values allowed are [in], [out], and [inout].

```
'lngTrecena [in]          Upper left corner trecena
'vein [in]                Upper left corner veintena
'lngRows [in]             Number of rows in the almanac
'alngBlack() [in]         Black distance numbers across almanac
'alngRed() [out]          Calculated Red trecena numbers across almanac
'aveinRowStart() [out]    Calculated Leftmost shown veintenas in almanac
'aveinAlmanac() [out]     Actual veintenas implied by almanac
'lngComplete [out]        Number almanac misses completing by.
'Return value:            None
```

Event procedures do not need a header unless the scope is changed to Public. Private procedures may need the header depending on the context. Note that the name of the routine is not referenced in the comments, making it possible to change the name of the procedure without changing the comments. No change history or coding history is included. Histories should be maintained by source code control systems, not by programmers since they are rarely properly kept up to date.

The comments are addressed to the person calling the procedure, and should include just enough information to tell the person how to call the procedure and use the returned values. After the On Error GoTo, other comments can be placed describing algorithms and other implementation details, if needed (although see the section on Comments).

Indenting

Tab stops should be set at four spaces. No member of a programming team should vary this number, as it makes editing other members of the team's code difficult.

All code inside a block should be indented one tab stop from the surrounding code, with exceptions noted elsewhere in this document. Indenting blocks makes finding the start and end of the block easy. A block is defined as the code that falls between the following keywords:

- Do/Loop
- Enum/End Enum
- For/Next
- For Each/Next
- Function/Exit Function/End Function
- If/Else/ElseIf/End If
- #If/#Else/#ElseIf/#End If
- Property/Exit Property/End Property
- Sub/Exit Sub/End Sub
- Type/End Type
- With/End With

For example:

```
For Each ci In tlio.Constants
    Set nodChild = tvw.Nodes.Add(Relative:=nod.Key, _
            Relationship:=tvwChild, _ Key:=ci.Guid & ci.Name,
            Text:=ci.Name, Image:=strcEnum)
         nodChild.EnsureVisible DoEvents If mboolShowProperties Then
             For Each mi In ci.Members
                 Set nodEnumChild = tvw.Nodes.Add(Relative:=nodChild.Key, _
                    Relationship:=tvwChild, Text:=mi.Name & strcEquals & _
                    mi.Value, Image:=strcConstant)
 nodEnumChild.EnsureVisible DoEvents Next mi End If Next ci
```

See also the section on Select/End Select Blocks.

{Alternative: The entire programming team may standardize on another number of spaces.}

{Alternative: Exit Function, Exit Property, and Exit Sub statements may be indented to the level of the surrounding code.}

Instantiation

An object variable should not be declared with New on the line it is declared on, unless there is a good reason to do so. The declaration should instead be broken into two lines. For example:

```
 Dim rst As ADODB.Recordset Set rst = New ADODB.Recordset
```

Not this:

```
 Dim rst As New ADODB.Recordset
```

Breaking it into two lines causes each reference to the rst variable to execute slightly faster. In addition, the object variable can be tested to see if it contains the value Nothing. For example:

```
 If rst Is Nothing Then MsgBox "rst not initialized" End If
```

If a one-line declaration is used, the above code would never execute the MsgBox because the reference to the rst variable in the If statement causes the object to be instantiated before the Is operator is evaluated. For Private and Public object variables, occasionally the convenience of using the New keyword outweighs the performance benefit, so the one-line declaration may still be used.

Labels

Labels in the code should be left justified, regardless of the indenting level of the surrounding code. They should appear on a line by themselves. For example:

```
ExitProcedure: On Error Resume Next
```

Long Lines of Code

VBA code editors will scroll a line of code to make the end visible. However, this makes it difficult to read the code quickly. It also means the code is not understandable if placed into a media that doesn't scroll, such as a paper print out or a book. For these reasons, the length of lines should be restricted.

A physical line of code should not exceed 80 characters. If a logical line of code exceeds 80 characters, then the line should be broken into two or more physical lines using the underscore line continuation character. All physical lines in the logical line following the first physical line should be indented one tab stop (four spaces) from the first physical line.

It may help to place a line at the top of the module with an apostrophe followed by 79 asterisks. Then the code window of the VBA editor can be sized to barely make the last asterisk visible. A fixed width font, such as Courier New, should be used to display the code in the VBA code window.

You should choose an appropriate place to break the line to enhance the maximum readability of the remaining code. When breaking lines that have a list separated by commas, you should break the line after a comma and before the next non-space character. For example:

```
Private Sub GetFiles(ByRef fso As Scripting.FileSystemObject, _ ByRef fld As
Scripting.Folder)
```

When breaking a line that is an expression built by operators, break the line before an operator of the expression. For example when the expression is built of string concatenation operators, break it like this:

```
 strParameters = strParameters & strAdd _ & strPassingConvention & pmi.Name & strArray
_ & strcAs & strDataType & strDefault
```

The next line becomes more readable this way.

If you have a long literal string, you may have to break the line like this:

```
 strValue = "This is a very, very long string that will cause the code " _ & "to wrap.
Because of this, you will need to break it."
```

In such cases, break it before the start of a word. Note that VBA performs the string concatenation at runtime, so this has performance considerations. In many cases, the string should be placed into a constant, an entry in resource file, or a database field and retrieved from there.

Comments should never be continued. When a comment exceeds 80 characters, continue the comment on the next line preceded by another apostrophe. See the section on comments.

Don't overly indent lines. Move overly indented code to a new procedure and call it from the original. In general, code should not need to be indented more than eight tab stops.

{Alternative: Place operators at the end of the line before the line continuation character instead of on the next line.}

Nothing

Explicitly set Object variables to Nothing before allowing the variable to be destroyed. This is especially true of object variables declared with the Dim keyword. For Example:

```
Public Sub Test(ByVal intValue As Integer)
```

```
 'error handling omitted for clarity Dim rst As ADODB.RecordSet Set rst = New
ADODB.RecordSet 'More code here Set rst = Nothing
```

```
End Sub
```

Setting the object variable to Nothing is not just good programming practice. If the rst object has code in its Class_Terminate event handler, that code can mess with global variables and objects.

In addition, set the object variable to Nothing is before exiting the procedure with Err.Raise. For example:

```
Private Sub Test(ByVal intValue As Integer)
```

```
 'error handling omitted for clarity Dim rst As ADODB.RecordSet Set rst = New
ADODB.RecordSet 'More code here If intValue > 1000 Then
```

```
 'Raise an Error----------------------------------------------------> Set rst =
Nothing On Error Goto 0 Call Err.Raise(Number:=lngcInvalidValue, _
```

```
 Description:=strcInvalidValue) End if Set rst = Nothing
```

```
End Sub
```

In the example just shown, if the rst object variable is not set to Nothing before performing the Err.Raise, the Class_Terminate of the rst object likely will change the properties of the Err object so that it no longer reflects the number given in lngcInvalidValue. This Class_Terminate code executes before the calling routine's error handler is invoked. This weird flow of execution has caused a number of very difficult to track down bugs.

Parameters to a Procedure

Every parameter to a procedure should be given an explicit data type, including variants. Every parameter should be passed by value using the ByVal keyword, with a few exceptions. These are:

- VBA doesn't allow certain data types to be passed by value, such as arrays, user-defined types, and objects.
- You specifically want to allow the changed value of the parameter to be passed back to the calling routine.
- The parameter to event procedure is specified as being by reference when VBA creates it.
- The arguments to a Declare statement must match the definition in the DLL.

Even in the cases where the argument should be passed by reference, you should explicitly prefix the parameter with ByRef, even though this is the default in VBA. This makes it explicit that you meant to pass that parameter by reference.

After VBA inserts an event procedure, the parameters to the event procedure should be changed to include ByVal and ByRef keywords, and change the parameter names to use the appropriate naming conventions. For example, VBA inserts the event procedure like this (with the line wrapped in this document):

```
Private Sub Form_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As
Single)
```

```
End Sub
```

This should be changed to read like this:

```
Private Sub Form_MouseMove(ByRef intButton As Integer, _ Byref intShift As Integer,
ByRef sngX As Single, ByRef sngY As Single)
```

```
End Sub
```

By changing it to read like this, the naming conventions indicate the data type and the ByRef keywords indicate that VBA may see the changes to the parameters.

Parentheses

Always use parentheses where the reading of the line may be unclear. For example, suppose that a line is written:

```
 If Not frmTest Is Nothing Then
```

It may not be clear that the Is operator has higher precedence than the Not operator in this line. Recode it to read:

```
 If Not (frmTest Is Nothing) Then
```

This makes it clear what order the operators are evaluated. The general rule is that if there is any question what the operator precedence is, use parentheses to make it clear.

Procedure Scope

Always use the Private scope on a procedure unless you need to expose the procedure outside the current module. In a library, use the Friend scope when you need a larger scope. Use Public only when access to the procedure is required outside the library. For Example:

```
Private Sub Test()
```

Project Properties [VB6]

In Visual Basic, the Project Properties dialog should always be filled in. These values may not apply to VBA hosts other than Visual Basic. Most of these fields can be retrieved from the EXE, DLL, or OCX file by right clicking on it in the Windows Explorer, then selecting Properties, then clicking on the Version tab in the dialog that appears. The values can be retrieved from within the program by getting properties of the App global system object. The following fields should be always be filled in:

- Project Name: The name of the library or standard EXE name. The library name should always start with a short word or abbreviation indicating the company or organization that is developing the library. For example, the Maya Calendar engine library from Xoc Software, might be named XocEngine or XocMayaEngine. This term is used for disambiguation of libraries and shows in the Object Browser. See the Disambiguation section. This is the internal name of the library. This may have abbreviations in it.
- Project Description: This should be the same name as the Project Name, except with spaces between the words. Abbreviations and the company or organization name should be spelled out. For example, use Xoc Maya Engine. These words show up in the VBA References dialog.
- Major/Minor/Revision number: These should be filled in with appropriate values. The version number should never be set to a smaller value as installation programs depend on it to determine if they should overwrite an older version with a newer one.

- Auto Increment: In most cases this should be checked. This automatically increments the revision number by one every time the project is compiled to a file.
- Application Title: The application title should be the name of the product that you expect to show externally, on the Windows Start menu, the Windows task list, the Windows Task Bar, and should be copied to the Caption of the main form in the application when the program starts. For example: Xoc Maya Engine.
- Comments: If Visual SourceSafe is used to maintain the project, this should be filled in with $Header: $. If keyword expansion in files is used, then Visual SourceSafe will place the expansion into the comments section of the executable. This gives the source name of the project is, the SourceSafe version number of the VBP file, the date and time the project file was changed, and by whom. This helps roll back the project to a given release to test for bugs. See the section on Source Code Control to configure SourceSafe.
- Company Name: Should be filled with your company or organization name. For example: Xoc Software. This is used on splash screens and about dialogs. Therefore, if your company is XYZ Software, Inc., you probably want to use XYZ Software.
- File Description: This is the description of how this file fits into the entire package. For example: Xoc Maya Calendar calculation engine or Xoc Maya Calendar UI.
- LegalCopyright: Enter the copyright notice for the program. For example: Copyright © 1999 by Xoc Software. You may find it useful to type Alt+0169 on the keypad (not the main keyboard) to get the © symbol in the dialog.
- LegalTrademarks: Enter any trademarks or registered trademarks for the company or product. For example: Xoc™ is a trademark of Xoc Software. You may find it useful to type Alt+0153 on the keypad (not the main keyboard) to get the ™ symbol and Alt+0174 to get the ® symbol. Note that these symbols may or may not show correctly in the application depending on the font you choose to display them.
- Product Name: This is the name of the product, without the company name. Therefore, if the name of the product elsewhere is Xoc Maya Calendar, the name here should be just Maya Calendar. This value may be used in splash screens and about dialogs.

Raising Errors

When you raise a runtime error from a component, to be trapped in the calling code, the error number that you raise should have a unique error number. For this purpose, VBA defines a constant vbObjectError that guarantees that errors that you generate will not conflict with ones that VBA defines. However, all libraries use errors in the range larger than vbObjectError, so you should strive to be different from the other libraries with your numbers. There is no way to guarantee this; the chances can be reduced by starting your errors at a random number in the range 512 to 32767 larger than vbObjectError. No library that an organization produces should ever have conflicting error numbers with another library from the same organization. For example: XYZ Software might start numbering its errors at vbObjectError + 4096. The first library produced from XYZ software might generate errors in the range from vbObjectError + 4096 to vbObjectError + 4146, the second library from vbObjectError + 4147 to vbObjectError + 4196, etc.

Select/End Select Blocks

The Select/End Select block is indented differently from other blocks (see Indenting). The Case blocks within the Select/End Select are lined up with the Select/End Select keywords. Code within a Case block is indented one tab stop from the Case statement. For Example:

```
Select Case Err.Number
Case tliErrCantLoadLibrary
    Err.Raise Number:=Err.Number, Description:=Err.Description, _
        Source:=Err.Source
Case 35602 'This key is already associated with an element of this
    'collection
    Set nodChild = tvw.Nodes.Item(cci.Guid)
    nodChild.Image = "InstClass"
    Resume NextItem
Case Else
       Call UnexpectedError(Err.Number, Err.Description, Err.Source, _
        Err.HelpFile, Err.HelpContext)
End Select
```

In non-RVBA coding standards, it is more common to indent Case blocks one tab stop from the surrounding Select/End Select. However, this causes the actual executing code to be indented two tab stops from the surrounding Select/End Select. The readability of the code is just as good, if not better with this scheme, although it takes some getting use to the first Case block being indented to the same level as the Select line.

See also the note about Case Else blocks in the section on Assertions.

{Alternative: Indent the Case blocks one tab stop from the surrounding Select/End Select. Then indent the code in the case blocks one more tab stop.}

Source Code Control [VB6]

Code should be maintained using some sort of Source Code Control. Microsoft Visual SourceSafe is the most common product used for this. When using Visual SourceSafe, the Administrator should configure it to expand keywords in files in the SourceSafe Administrator Options dialog. The following files should be expanded: *.bas,*.cls,*.ctl,*.frm,*.pag,*.vbp. Entries such as $Header: $ can then be placed into the code and are expanded automatically. See the SourceSafe documentation on keyword expansion. Also, see the use of the Comments entry in the section on Project Properties in this document.

Type Conversion

VBA is considered a weakly typed language. You can construct expressions such as this one:

```
strValue = "Your order came to " & intQuantity * curPrice
```

VBA will automatically convert the result of the expression into a string to make the expression work. However, it is better programming practice to make explicit what VBA is doing using the type conversion functions: CBool, CInt, CLng, CStr, etc., plus the Format$ function. For example:

```
 strValue = "Your order came to " _ & Format$(CCur(intQuantity) * curPrice,
"$#,###.00")
```

The RVBA naming conventions will help point out possible bugs. If you see a line that looks like this, you may have a potential bug:

```
intValue = lngInput
```

If the value in the variable lngInput is 90,000, this line will cause an Overflow runtime error. The fact that the types of the variables are different is a clear warning sign. If, however, you knew the value in lngInput could only be in the range 1 to 1000, it might be acceptable to do the assignment like this:

```
 Debug.Assert lngValue >= 1 And Debug.Assert lngValue <= 1000 intValue =
CLng(lngInput)
```

See also the section on Assertions.

Variable Declaration

Every variable should be explicitly declared. Using the Option Explicit keyword at the top of the module will have VBA enforce that. The VBA editor's Tools Options dialog has a setting that will make this be automatically inserted in all new modules.

Every variable should be given an explicit data type. This includes variants, which are the default. For example, a variant should be declared as:

```
 Dim varValue As Variant
```

Rather than letting it be implicitly defined or declaring it as:

```
 Dim varValue
```

Every variable should be declared on a line by itself. This precludes running into this bug:

```
 Dim intValue, intTest As Integer
```

That declaration makes it obvious that the two variables were meant to be declared as integers, but the first variable is defined as a variant. If instead the declarations were made on a line by themselves, the problem goes away. For example:

```
 Dim intValue As Integer Dim intTest As Integer
```

In addition, by declaring each variable on a line by itself, you can use the Ctrl+Y keyboard shortcut to cut the declaration to the clipboard regardless of where the caret is on the line, then paste it somewhere else. If there are multiple declarations on a line then editing is not as easy.

Variable Initialization

Some languages allow declaring a variable and giving it an initial value at the same time. Visual Basic doesn't allow that. A variable has a default value at the time that it is declared based on its data type. However, a syntax variation can be used inside of a procedure to give the feel of initializing it and assigning the default value.

Visual Basic allows you to place multiple logical lines of code on the same physical line if you separate them with colons. It also allows you to mix the declaration of variables with executable lines of code. Therefore, inside a procedure you can initialize a variable and give it a default value in one physical line like this:

```
Public Sub Test()
     'Error handling omitted for clarity
     Dim intValue As Integer: intValue = 7
     Dim strTest As String: strTest = "Default Value"
     'other code End
Sub
```

Variables Scope and Lifetime

Variables should always be declared with the smallest level of scope and the shortest lifetime possible. Thus, you should declare variables with Dim inside of procedures by preference. If you need a longer lifetime, then use Static. If you need a wider scope, use Private. Only use Public as a last resort. Public variables declared in standard modules are global and can be changed by any piece of code throughout the entire project. This makes debugging changes to their value very difficult. Global variables of this sort should only be used in the context where they are set during initialization of the program, then remain static for the rest of the time the program executing.

Global variables should never be changed in one part of the program to be retrieved in another part of the program. In such cases, you should use parameters of procedures or properties of forms or objects to pass the information. If there are more than 20 global variables in the program, it is a warning sign that the program design is wrong.

Having Public constants are allowed and encouraged. See the section on constants.

Version Compatibility [VB6]

After building an ActiveX control or ActiveX DLL for the first time in Visual Basic, the project compatibility should be set to Binary Compatibility in the Project Properties dialog. Each time the component is "released," a copy of the component should be made to the same directory, but with the filename extension set to CMP. The binary compatible file entry should point to this file. That means that you can modify the interface to the file within a release as long as you are still backwards compatible with the last release. The CMP file should be checked into the Source Code Control project, whereas the current copy of the component itself probably should not be checked in (see the section on Source Code Control).

Greg Reddick is the President of Xoc Software, a software development company developing programs in Visual Basic, Microsoft Access, C/C++, and for the web. He leads training seminars in Visual Basic for Application Developers Training Company (AppDev). In a previous life, he worked for four years on the Access development team at Microsoft. Greg can be reached at mailto:grr@xoc.net or from the Xoc Software web site, http://www.xoc.net.

APPENDIX E: MICROSOFT APPLICATION USER INTERFACE GUIDELINES

Intro

By using established guidelines we can ensure that we build our software products with appropriate and consistent appearances.

We will follow the standard Windows user interface guidelines as described in "*Windows User Interface Guide*", Microsoft Press, 1999, and as described in the three documents that can be found at http://sdg.jsi.com/standards/style-guides/user-interface/windows. This document further specifies the guidelines set forth in those documents.

General Properties

Some properties are common to pages, controls, and elements. The rules set forth in this section should suffice in most cases.

Fonts

Face

When not defaulting to the system font, use Arial.

Size

Text

Use 8 point, normal.

Customer/Client

Use 12 point, normal.

Page Title

Use 14 point, bold.

Type

Use boldface type for titles. Use normal type elsewhere. Never use underlined or italic typefaces.

Color

All text should be black.

Background Color

The font background should always be transparent, thereby defaulting to the background color of the font's parent form or control.

Case

Use title case for all labels, names, and titles. Do not use either all UPPERCASE or all lowercase.

Color

Palette

Deliver has selected a palette of five colors: red, orange, gold, green, and blue.

Red	HSL: 234, 195, 85 PMS: 194 RGB (decimal): 164, 17, 40 RGB (hexadecimal): #A41128 Windows: 2625956	🟥
Orange	HSL: 20, 40, 120 PMS: 152 RGB (decimal): 255,125,0 RGB (hexadecimal): #FF7D00 Windows: 32255	🟧
Gold	HSL: 25, 240, 129 PMS: 137 RGB (decimal): 255, 168, 20 RGB (hexadecimal): #FFA814 Windows: 1353983	🟨
Green	HSL: 74, 143, 74 PMS: 363 RGB (decimal): 47, 126, 32 RGB (hexadecimal): #2F7E20 Windows: 2129455	🟩
Blue	HSL: 138, 229, 64 PMS: 541 RGB (decimal): 3, 76, 133 RGB (hexadecimal): #034C85 Windows: 8735747	🟦

Text

All text should be black. For further info on text properties, see the section on "Fonts".

Dimensions

Screen

Height

All screens will fit onto a medium resolution terminal, i.e., 600 pixels high. If at all possible, no data will be displayed below the viewable area of the screen. In other words, avoid vertical scrolling.

Width

All screens will fit onto a medium resolution terminal, i.e., 800 pixels wide. No data should ever be displayed to the right of the viewable area of the screen. In other words, horizontal scrolling is forbidden.

Regions

Header & Primary Navigation

The header and primary navigation area will span the entire width of the page, occupying no more than the top 75 pixels of the screens height.

Secondary Navigation

The secondary navigation area will run down the left side of the screen, below the header. Of this space, the secondary navigation bar will occupy no more than 15% of the screen's width.

Content

The content area will fill the area from the bottom of the header to the top of the footer, and from the right edge of the secondary navigation area to the right edge of the screen.

Ordinarily, this region will be subdivided into other regions dictated by the application and the application's purpose.

Footer

The footer will span the breadth of the page, occupying no more than the bottom 40 pixels of the screen's height.

Controls

Controls should be of uniform dimensions.

Buttons

(Borrowed from GUI LNF Standards – DENR (Interact))

If the length of text for a series of command buttons in a dialog box is similar, make all the buttons in the dialog box the size of the largest button

If the text length for a series of command buttons in a dialog box varies, use two button sizes—one for shorter text and another for longer text. Do not use more than two different button sizes in a dialog box.

Text Boxes

(Borrowed from GUI LNF Standards – DENR (Interact))

Size text boxes to indicate the approximate length of the field. If you have text boxes of similar length, make them the same length unless you need to show the exact size of the field. If the length of the field can vary, use text boxes of the same length to minimize the number of unique margins on the screen.

Left align text boxes on the screen to minimize the number of different margins. If a particular text box has a long label, use a different margin for that text box. Limit the number of unique margins to two.

List Boxes
Show at least three, but no more than eight items in a list box at a time. If you have more items use a scroll bar to view the rest of the items.

Flow
Tabbing from control to control will go from top to bottom, then left to right.

Usability

Keyboard vs. Mouse
All applications will be fully usable without a mouse, i.e., all functionality will be readily accessible from the keyboard.

Section 508
All applications will be Section 508 compliant.

Performance
The system will always provide some visual feedback to the user as soon as possible. For desktop applications, this feedback must occur within one second. For web applications, the feedback must occur within five seconds.

Internationalization

Multi-Language support
All apps will provide Unicode (wide character) multi-language support. This may mean that dialogs & screens have to be somewhat auto-sizing.

Date & Time
All applications will use the system-defined date and time formats.

Applications
Aside from the general, overarching principles used to govern screen layout and design, the various types of applications have their own special constraints.

Desktop
Desktop applications are applications that can be run in standalone mode without the use of a browser. These are traditional Windows applications.

Page
Splash Screen
Each Deliver app will feature a splash screen of uniform layout (actual layout is to be determined by the communications group). This splash screen will be 300 pixels by 400 pixels. The splash screen will persist for 5 seconds. The splash screen will be available from Help → About <appname>.

Framework
Each application will feature a title bar, a menu bar, a tool bar, and a status bar.

(Microsoft Access-based applications will use the framework provided by Access.)

Title Bar
The title bar is the line at the very top of the application or dialog, just above the menu bar (if the menu bar is present). The title bar contains the minimize, maximize, and restore buttons.

Font
The font should be Arial 12, bold.

Color
The title bar should be either the system color, or the appropriate color from the Deliver palette. Which one?

Content
The title bar should read "Page Name – Application Name".

Case
Titles should be in Proper Case, except in the case of logos.

Menu Bar
Standard menu bar names and positions will be used.

Tool Bars
Tool bars will use standard Windows icons.

Icon Size
There are two sizes, small and large. Small is 16x16 pixels, large is 20x20 pixels. Prefer the small icons.

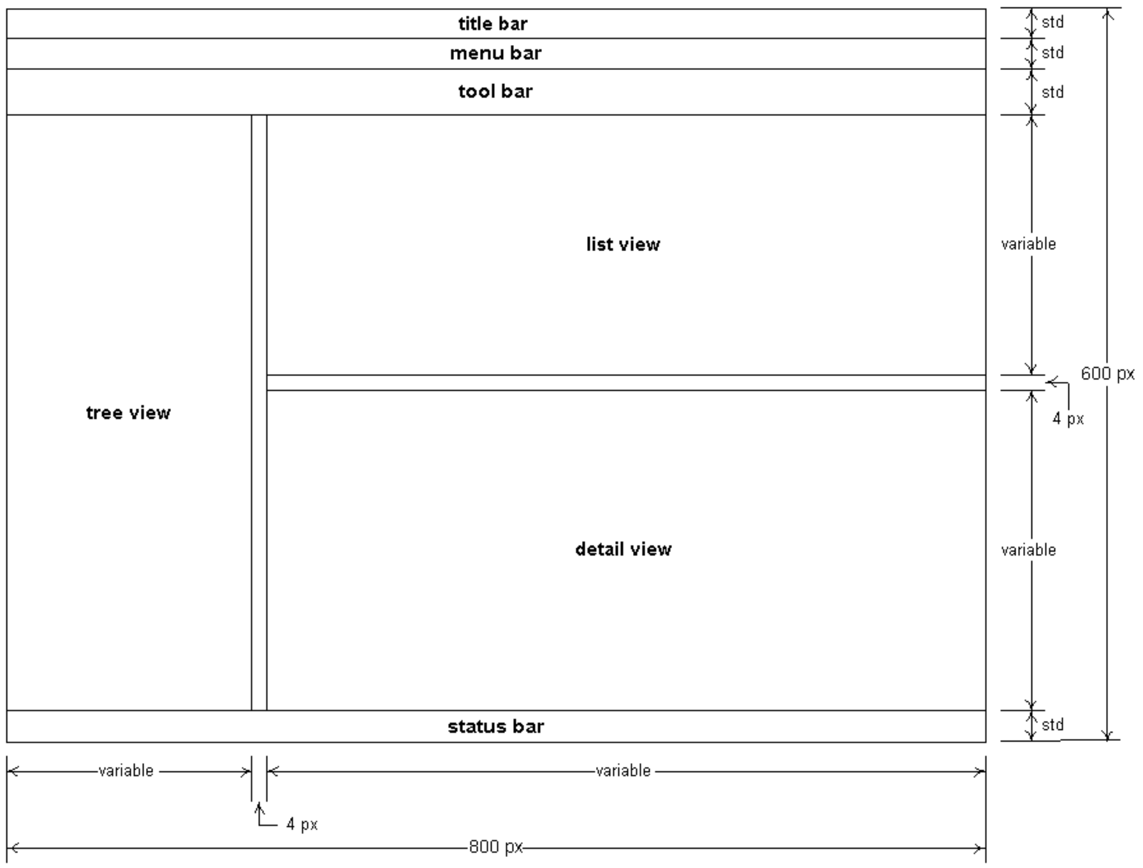
Spacing
For each size, there should be 3 pixels between a toolbar button and its text label.

Status Bar
The status bar will be the Windows standard 1 line tall. It should be used to provide the user with information about the status of the application.

Sizing
All screens will support minimizing, maximizing, and custom sizing.

Layout
General
Deliver uses "explorer" style applications. This provides for easiest navigation of the application's screens.

The following drawing defines the recognized screen regions and their sizes.



Size

Target screen resolution: all apps should be presentable at 600 x 800.

Each pane in the window can be resized.

Windows will control the heights of standard Windows elements such as title bar, menu bar, tool bar, and status bar.

Font

Windows will control the fonts in the title bar, menu bar, tool bar, and status bar.

Color

Windows will control the colors in the title bar, menu bar, tool bar, and status bar.

Graphics

Splash Screen

Has the app logo, product version info, JSI logo, funding source (customer) logo, and application image.

About

Use the same info as the splash screen, add links to websites, email address.

Regions

The screen is divided into number of regions, each of which has a purpose.

Tree View

The Tree View is used to navigate the application.

Icons
We will use the following standard icons for the following purposes:

Form
Forms will be represented by a little picture of a form. (need sample)

Report
Reports will be represented by a piece of paper with writing on it. (need sample)

Question
Question marks are commonly used for this. (need sample)

Configuration
A picture of tools will represent configuration screens. (need sample)

Group/Section
This could be either plus sign for closed groups, and a minus sign for open groups, or opened and closed folders. Either set of icons is commonly used in Windows apps. (need sample)

Forms
Forms are essentially dialogs that do not pop up, but remain embedded in the application framework. Whereas dialogs will contain a title bar and minimize/maximize/close buttons, a forms do not.

Layout
For controls that do not contain their own labels, the label should be placed to the left or above the related control. This makes it easier for users to associate the label with the corresponding control.

The logo should go in the upper right corner.

All forms will support resizing, minimizing, and maximizing.

Logos
All Deliver sw has a logo area in top right area of each screen. This logo area will be used to brand the product as a Deliver app. This area will likely contain only the application's logo. The JSI and customer logos would appear on the splash screen and in the "About" dialog.

Style (Special Effect)
Dialogs and forms should be flat.

Elements
Screen elements are the things that go in the regions. These are things like tool bars, labels, and titles.

Title Bar
This is the same as standard definition for a title bar, except that the dialog title bar should not include the " – Application Name".

Title
If a title is present it should be bold, 12pt, default font, proper case. No italics; no underline.

Menu Bar
Use standard menu bar.

Tool Bar
Use standard tool bar.

Labels
Use the default font.

Pop Up Dialogs

Modal

When to use

Modal dialogs are used whenever the application absolutely cannot continue without user input. In most cases, though, this can be handled with an "Apply" button.

Buttons

As a minimum, modal dialogs will have "OK", "Cancel", and "Help" buttons.

Modeless

When to use

Modeless dialogs are to be used unless the criteria for using a modal dialog are met.

Buttons

As a minimum, modeless dialogs will have "OK", "Cancel", and "Help" buttons. "Apply" is almost meaningless – the user must hit either "OK" or "Cancel" to close the dialog, and the functionality of "OK" is a superset of that of "Apply".

Tabbed

When to use

Use tabbed dialogs when the dialog uses more controls than can be fit on a single screen.

Layout

Each tab should contain controls that relate to a particular dialog subtopic. If a control does not pertain to the topic of the overall dialog, then it should not be included on any of the tabs.

Buttons

Use consistent tab width, allowing the longest tab label dictate the tab width.

Each page of the tabbed dialog will feature, as a minimum, "OK", "Cancel", and "Apply" buttons. The buttons will follow the properties set forth in the General Properties section of this document.

Elements

Title Bar

Same as standard definition for a title bar, except that the dialog title bar should not include the " – Application Name".

Title

If a title is present it should be bold, 12pt, default font, proper case. No italics; no underline.

Menu Bar

Do not use menu bars in dialogs except in exceptional circumstances.

Tool Bar

Do not use tool bars on dialogs except in exceptional circumstances.

Labels

Use the default font.

Style (Special Effect)

Dialogs should be flat.

Reports

Size

DP-Body textFormat

Paper Size

DP-Body textMargins

DP-Body textLayout

DP-Body text

Controls

DP-Body textGeneral

Dialog Base Units

DP-Body textSize

DP-Body textExamples:

DP-Body textSpacing

DP-Body textControls

DP-Body textThe absolute smallest space between controls is 2 DLUs.

Dialogs

There should be a 7 DLUs between the edge of the dialog box and the text or frame.

Paragraphs

There should be 7 DLUs between paragraphs of text.

Text Labels

There should be 3 DLUs between text labels and their controls.

Group Boxes

The first control in a group box should be 11 DLUs down from the top of said group box.

Controls in a group box should be aligned vertically to the group box title.

The last control in a group box should be 7 DLUs above the bottom of the group box.

Buttons

If a text label is beside a button, it should be 3 DLUs down from the top of the button.

A check box, list box, or option button beside a button should be 2 DLUs down from the top of the button.

Grouping

The following rules apply to grouping:

- Group related components
- Group box controls
- Use separator lines on menus

Much more comprehensive discussions of grouping are available in the resources cited in the introduction to this document.

Color

Color can be used, but is not recommended. Allow the user to change color schemes.

Other considerations

Main command buttons in a secondary window should be stacked in the upper right corner or in a row along the bottom. If there is a default button, it should always be the first one in the set. OK and Cancel buttons should be placed next to each other.

Alignment
In group boxes, controls should be left-aligned with the text label of the group.

Command buttons in the group should be right-aligned.

In toolbar arrangements, buttons and other controls are typically left-aligned or top-aligned

Text box
Layout
Use a consistent width between boxes.   Flow should be from top to bottom, left to right.

Unlocked
Font
Use the standard default.

Color
Text
Use the standard text color.

Background
White is the standard Windows color for the background of unlocked text boxes. We will let the system control this color.

Size
Text boxes should be of uniform length, unless there is a requirement to show the user how large the data field is.

Locked
Font
Use Deliver default.

Color
Text
Use the standard text color.

Background
The Windows standard color for locked text boxes is light gray, but we will let the system control this color.

Size
Text boxes should be of uniform length, unless there is a requirement to show the user how large the data field is.

List box
Font
Use default.

Color
Use default.

Size
Display from 3 to 8 rows. Allow vertical scrolling if there are more than 8 items on the list.

Avoid horizontal scrolling.

Combo box
Font
Use default.

Color
Use default.

Size
Display from 3 to 8 rows. Allow vertical scrolling if there are more than 8 items on the list.

Avoid horizontal scrolling.

Radio buttons
When to use
Use option buttons when users should pick one mutually exclusive choice from a list of options, for example, choosing a pay period in a personnel application.

Colors
Let the system decide.

Size
Use default.

Arrangement
Lay these out vertically, using an outline to group them.

Check boxes
When to use
Use check boxes when users can choose one or more options, but these choices are not mutually exclusive.

Use check boxes for toggling a single value on or off. It is okay to have just one check box.

Colors
Let the system decide.

Size
Use default.

Arrangement
Lay these out vertically, using an outline to group them.

Command Buttons
Size
See the description of Buttons in General/Dimensions/Buttons.

Behavior
Ok
Saves settings and closes screen.

Cancel
Discards changes and closes screen.

Apply
Saves changes and keeps user on current screen.

Clear
Discards changes and keeps user on current screen.

Next
Takes user to the next logical screen. Does not implicitly save changes.

Previous
Takes user to the next logical screen. Does not implicitly save changes.

Help
Displays help text.

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