J Street LIMS

Laboratory **I**nformation **M**anagement **S**ystem

for Microsoft Access

Programmer's Guide

Version 6 for SQL Server

J Street TECHNOLOGY

J Street Technology
16625 Redmond Way Ste M
PMB 463
Redmond, WA 98052
425-679-6206
www.JStreetLIMS.com

Table of Contents

Chapter 1: Introduction

J STREET LIMS is a Windows-based laboratory information management system (LIMS) used to manage the daily information processing requirements of an analytical laboratory. From sample login to final report generation, J STREET LIMS will help organize, track, and manage your data. This document provides an overview of the system's architecture along with guidelines for customizing the system's software.

Programmer's Guide Overview

This Programmer's Guide is intended for software development personnel who plan on customizing the system's software and database objects. This document assumes that you are familiar with Microsoft Access, Visual Basic for Applications (VBA) development, and SQL Server. If you are new to these applications, you should refer to the additional documents listed below. This guide is organized into the following chapters:

Chapter 2: Getting Started
This chapter provides an overview of key system components, programming and naming conventions, and guidelines for customizing the system.

Chapter 3: Tables
This chapter outlines the types of database tables used in the system and provides a detailed description of the system's internal configuration tables.

Chapter 4: Queries
This chapter describes the programming conventions used to implement the system's queries.

Chapter 5: Forms
This chapter describes the programming conventions used to implement the system's data entry/inquiry forms.

Chapter 6: Reports
This chapter outlines programming conventions used within the system's reports.

Chapter 7: Macros
This chapter provides an overview of the system's macros.

Chapter 8: Modules
This chapter outlines programming conventions used within the system's modules.

Additional Documents

Refer to the following documents for additional information on J STREET LIMS and the Access development environment.

Document	Author
J STREET LIMS Release Notes	J Street Technology
J STREET LIMS Knowledge Base at https://www.JStreetLIMS.com	J Street Technology
J STREET LIMS User's Guide	J Street Technology

System Overview

*See the **J STREET LIMS Release Notes** for installation instructions.*

See the J Street LIMS Installation Guide for System Requirements and installation instructions

Editing LimsCode

With a J STREET LIMS Full System license and a full copy of Microsoft Access you can customize the J STREET LIMS software by modifying your .aacdb file. To edit, start Access then open the file using the Shift bypass key (i.e. hold down the Shift key while clicking Open) to preserve the Access ribbon. Double-click frmMainMenu to run the SystemStartup procedure and display the main menu. Resize the Access window to allow LIMS screens to display in their entirety along with the Access Objects list. The LIMS will preserve the screen size and resolution for this developer mode in the .xml file in the same folder as .aacdb file.

Creating LimsCode.accde

The J STREET LIMS software (LimsCode) is installed and runs on each LIMS workstation using .accde files rather than .accdb files. ACCDE files are databases created from ACCDB files but with all Visual Basic for Applications (VBA) source code removed. Only the compiled VBA code is stored in an ACCDE file making the file smaller than the ACCDB file. Without any source code, an ACCDE file cannot become uncompiled. Uncompiled ACCDB files will run more slowly than its compiled ACCDE version.

If you have an J STREET LIMS Full System license you will find the ACCDB files with all VBA source code for LimsCode6 and LimsMapi6 in the ACCDB folder within the Setup folder containing the J STREET LIMS installation files. Using a full copy of Access, you can customize your system by opening LimsCode6.accdb (see Editing LimsCode above). After you have made your changes to LimsCode6.accdb you need to create LimsCode6.accde and distribute the ACCDE file to all J STREET LIMS workstations. To create the ACCDE file, close all open objects (forms, reports, etc.) then double-click any module to open the VBA editor. Use Debug | Compile LimsCode6 to check for syntax errors. If no errors are found, use File | Close and Return to Microsoft Access to close the VBA editor and return to Access. On the File tab use Save As then click the Make ACCDE advanced option. Select the folder where file LimsCode6.accde should be saved then click [Save] to create the file.

TIP
When creating LimsCode6.accde, save the file directly to a file server folder accessible from all LIMS workstations, and then use the options on the LimsCode tab of the System Configuration screen in J STREET LIMS to automatically distribute the updated file to each workstation.

Chapter 2: Getting Started

This chapter provides an overview of key system components, programming and naming conventions, and guidelines for customizing the system.

System Overview

J STREET LIMS is an all-Access application that uses split database architecture. LimsCode, the front-end database, contains all forms, queries, reports, macros and modules. The J STREET LIMS SQL Server back-end database contains tables, views, and stored procedures and is the repository for all user-entered data. With this architecture, the front-end database is installed on all LIMS workstations. In a single-user configuration, the SQL Server database resides on the same workstation as LimsCode. In multi-user environments, the SQL Server database resides on either a dedicated file server or one workstation acting as the LIMS server in a peer-to-peer network. In both single- and multi-user settings, LimsCode *attaches* the SQL Server back-end database. In multi-user environments this split database architecture provides optimal performance since only data is transferred across the network. All forms, reports, and software reside on the local workstation. Due to the use of LimsCode temporary data tables, LimsCode can not be shared on a server and must reside on each LIMS workstation.

NOTE

While network installations of Microsoft Office are often used to simplify a network administrator's tasks when installing and maintaining software applications, a network installation of Office is not recommended with J STREET LIMS. The main benefit of a split Access database architecture is to maximize front-end performance and limit network traffic to back-end database access. The additional network traffic required to load Office executables from the server can result in a noticeably slower system.

To open LimsCode in the Access development environment, start Access, use Open on the File tab then navigate to the folder where LimsCode6.accdb has been copied. Hold down the Shift key while double-clicking LimsCode6.accdb. If LimsCode6.accdb is already in your list of recent files on the File tab, simply hold down the Shift key while clicking the file.

After opening LimsCode in the Access development environment, you may find it helpful to browse the list of database objects under each of the navigation pane's options: **Tables**, **Queries**, **Forms**, **Reports**, **Macros**, and **Modules**.

Naming Conventions

Following is a list of the database object and control naming conventions. Use of the listed prefix for control and object names allows quick identification of the control or object type.

Objects

frm	Form
qry	Query
rpt	Report
sys	LimsCode tables (non-attached)
usp	SQL Server stored procedure
v	SQL Server view
z	internal/development (not part of shipped system)

Controls

btn	Option Button
cbx	Combo Box
chk	Check Box
cmd	Command Button
frm	Subform
lbl	Label
lst	List Box
ole	OLE Object
opt	Option Group
txt	Text Box

Programming Conventions

Following are naming and other conventions used in the VBA code in modules and code behind forms and reports.

Note that these conventions apply to LIMS-specific code, but not the code library routines and procedures introduced in version 6 of J STREET LIMS. Refer to "Version 6 Changes" in the Modules chapter for more information.

Variable Names

The following variable prefixes are used to identify the variable's type and scope. When a scope identifier is used it precedes the type identifier (e.g.

gsLimsMsgs is a string variable with global scope). Note that all modules use VBA's Option Explicit statement to force explicit declaration of all variables within the module.

b	Boolean
c	Const
d	Double (floating pt.)
dt	Date
n	Integer, Long (and old style boolean)
o	Object
s	String
str	String (old style)
v	Variant
con	Container
ctl	Control
db	Database
doc	Document
fld	Field
qry	QueryDef
rs	RecordSet
tbl	TableDef
ws	WorkSpace
m	module scope
g	global scope
ALL_CAPS	old style Const

Comments

Most modules, including code behind forms and reports, include a single comment block with a short module description and a record of significant changes. Minor changes are noted using comments within a procedure. The version number and developer's initials should be added to the revision history. Following is an example of a module comment block.

```
' Module Name.... frmSampleQuery
' Description.... Standard sample query dialog (used as subform)
' Note........... Calling form should set the value of
'                 txtWhichSamples to control the form's mode.
' Revisions...... 091197 rdc 1.6.0  Original
'                 051498 zz  1.7.0  replace txtSQL with cbxSQL
'                 052798 zz  1.7.0  Customer access control
'                 111698 rdc 2.0.0  Added multi-select list boxes;
'                                   added txtSQL over cbxSQL to
'                                   solve truncating problem.
```

Minor code changes include the developer's initials and version number in comment lines and inline comments:

```
' RDC 1.6.2: use two passes.  First look for location-specific
' analyses/requirements. If nothing is found, use
' the project's default analyses/requirements.

ws.BeginTrans               ' RDC 1.6.1
```

Inline comments are also used to help document lengthy nested programming constructs:

```
        Loop    ' until rs.eof
    End If      ' if rs.nomatch else
    rsRow.MoveNext
Loop            ' until rsRow.eof
```

Error Handling

J STREET LIMS uses standard error handling in every VBA procedure. Any customizations should continue to follow this convention. Proper error handling is important to provide the user a useful error message and to prevent program halts when using the Access runtime. All procedures should trap runtime errors and use J STREET LIMS' standard ErrorLog procedure. The ErrorLog procedure displays a MsgBox with the error description and logs the error information to the back-end's ErrorLog table. Following is the structure of a typical VBA procedure using proper error handling and the ErrorLog procedure.

```
Private Sub cmdQuery_Click()

    On Error GoTo cmdQuery_ClickErr
    '
    ' procedure's code goes here
    '
    Exit Sub

cmdQuery_ClickErr:
    ErrorLog Me.Name & ".cmdQuery_Click", Error$
    Exit Sub

End Sub
```

The ErrorLog procedure requires two parameters: the procedure name and the error message. In the form module example above the form name (Me.Name) and procedure name are concatenated and passed to the ErrorLog procedure so we know exactly where the error occurred.

Bound Forms

So that a better and more informative message than the native ODBC error can be displayed, all forms bound to back-end tables require the following conventions:

1. Add "Required=fieldname;" to Tag property of any control bound to a backend required field

2. Add/update Form_BeforeUpdate and include call to LIMSFormBeforeUpdate
3. Add/update Form_Delete and include call to LIMSFormDelete
4. Add/update Form_Error and include call to LIMSFormError
5. Add "DeleteError=Related tables include …" to the form Tag property of any form where we want to give a hint which tables may be preventing the delete.
6. Add "UpdateError=…" to the form Tag property of any form where we want to identify which fields must be unique.

For example, below is the Tag property of frmSetupLocations:

```
DeleteError=Related tables include Sample, Customer Sample
Login, Login Batch Sample, Project Analysis, and Project
Requirement.; UpdateError=The location name must be
unique.;
```

And following is the Tag property of control txtName on frmSetupLocations:

```
Required=Location;
```

The Popup Calendar

To speed entry of dates, J STREET LIMS includes a popup calendar for easy date selection. Double-clicking within any date field opens the calendar. If a date is selected it is inserted into the field when the calendar is closed. This capability can be added to any text box used for dates by adding a few lines of code to the text box's double-click event procedure. Following is an example.

```
Private Sub txtCollectedDate_DblClick(Cancel As Integer)

    On Error GoTo txtCollectedDate_DblClickErr
    Dim vDate As Variant

    vDate = PopupCalendar(txtCollectedDate)
    If IsDate(vDate) Then txtCollectedDate = vDate
    Exit Sub

txtCollectedDate_DblClickErr:
    ErrorLog Me.Name & ".txtCollectedDate_DblClick", Error$
    Exit Sub

End Sub
```

Global Values

*Double-click **frmGlobalValues** in the list of forms to unhide the form.*

J STREET LIMS uses a global values form (frmGlobalValues) to maintain system information and to support communication between forms and reports. The global values form is opened and hidden during system startup and remains open throughout the LIMS session. The form's data is then available to all VBA procedures at all times. Unhiding the form during development sessions is a valuable debugging tool since all of the form's data is easily viewed. The global values form's underlying query obtains data from LimsCode's sysGlobalValues and sysWorkstationConfiguration tables and the back-end database's SystemConfiguration table.

Customizing the global values form is not recommended since it is often updated with new versions. If similar capabilities are necessary, consider adding a second global values form for site-specific customizations. Alternatively, global variables in a custom module can also be used.

System Startup

While Access supports database startup forms, a more flexible approach is to use a startup VBA procedure which is how J STREET LIMS startup tasks are performed. LimsCode's AutoExec macro includes a single RunCode action which runs the SystemStartup function in the LIMS Functions module. This single procedure performs all of the following startup tasks.

1. Check for annual subscription expiration.
2. Display the splash form which shows the copyright notice and licensee information. The form will be closed below.
3. Make sure LimsCode is not read-only. If LimsCode's read-only attribute is set, the system exits with a message. LimsCode cannot be a read-only file since data must be written to 'sys' tables.
4. Load the workstation configuration settings. Tables sysWorkstationConfiguration and sysWorkstationPrinter are initialized with the contents of an XML file (created by frmGlobalValues.Form_Close) in the same folder and with the same name as LimsCode (e.g. LimsCode6.xml). If the XML file does not exist, records with default values are added to the 'sys' tables.
5. If the LimsCode size warning has not been suppressed in the Workstation Configuration screen, determine the size of the LimsCode database. If the size exceeds the set limit, issue a warning message. If sysWorkstationConfiguration's CompactOnClose or DeleteTempRecsOnClose options are not set, the system offers to turn them on.
6. If the last attached SQL Server database is no longer available, an Open LIMS Database dialog allows the user to select a valid back-end database.
7. Check for an updated LimsCode. Using the options on the LimsCode tab of the System Configuration screen, the LIMS administrator can distribute an updated LimsCode to all LIMS workstations by placing the file in a shared folder, updating the file's revision date and time, and

enabling the automatic update feature. If the automatic update is enabled and a newer LimsCode is available, the user is given the option to install the update. If the user selects to install the update, a BAT file is created to perform the copy then the system uses Shell to execute the BAT file and exits.

8. Issue a warning if the workstation has no default Windows printer.
9. Enable or disable Adaptive Menus based on the setting in sysWorkstationConfiguration.
10. Set the database's HyperlinkBase property using the setting in sysWorkstationConfiguration.
11. Verify the number of licenses is not exceeded.
12. Open and hide the global values form.
13. Set the multi-user options specified in the System Configuration screen: update retry interval and number of update retries.
14. Set miscellaneous global variables by checking SysCfg records in the back-end database's MiscellaneousList table.
15. Verify the compiled state of LimsCode's VBA project. If the VBA project is not fully compiled and the warning has not been supressed in the Workstation Configuration screen, issue a warning message.
16. Set the application's icon to file J Street LIMS5.ico located in the same folder as LimsCode.
17. Copy the MainMenu table's contents from the SQL Server database to sysMainMenu.
18. If LimsCode is an ACCDE file and the user is a member of the Admins role, disable the Admin menu's Install Version Update and Recompile options.
19. Using the previous session's settings in sysWorkstationConfiguration, restore the window size and position then close the splash form.
20. If we are in production mode (Access runtime or ribbon is not visible) open the main menu.
21. If we are in production mode and the Workstation Configuration screen's "Check sample schedules at startup" option is enabled, look for any schedule warnings and display the schedule warnings screen if any are found.
22. If we are in production mode and the Workstation Configuration screen's "Check sample warnings at startup" option is enabled, look for any sample warnings and display the sample warnings screen if any are found.

Modifying the SystemStartup procedure is not recommended since it may be modified in future version updates. If site-specific startup tasks must be included, add a second RunCode action to the AutoExec macro to invoke a custom startup procedure added to a custom module.

Custom Ribbon and Shortcut Menu Bars

J STREET LIMS includes a number of custom ribbons that are used throughout the system. LimsCode ribbons are generated in code within the jlRibbon module.

Customizing LimsCode's built-in ribbon tabs is not recommended since they are subject to change with version updates. When additional custom features are necessary, create additional ribbon tabs and add them to the main ribbon via the **BuildRibbons** subroutine in jlRibbon. While the ribbon interface is more powerful and flexible than the older-style menus and toolbars, it is also more difficult to customize. Contact us if you need help migrating custom toolbars and menu bars from versions 4 or 5 to version 6.

There are three types of ribbons in Microsoft Access:

1. **Application Ribbon**: the default ribbon that appears for all forms and reports
2. **Backstage Ribbon**: the menu that appears when you click File in the top left of the application
3. **Context Ribbons**: ribbons that appear only when certain forms or reports are active

There are two primary ways to customize the existing ribbons in J STREET LIMS:

- Add one or more tabs to the Application Ribbon
- Add one or more Context Ribbons and assign them to form or report objects via the RibbonName property

Refer to the existing code in the jlRibbon module for implementation details.

Following is a list of custom ribbons in J STREET LIMS:

gbRibbon_MasterRibbon	This is the main application ribbon. The backstage ribbon is defined within this named ribbon.
gbRibbonPrintPreview	All reports display this ribbon in the print preview window.
gbRibbonByAnalyte	The results by analyte form uses this ribbon which includes a Spreadsheet tab.

gbRibbonBySample	The results by sample form uses this ribbon which includes a Spreadsheet tab.
gbRibbonBatchLogin	The batch login form uses this ribbon which includes a Spreadsheet tab.

J STREET LIMS also includes a number of custom shortcut menus. Use the shortcut menu bar property of a form control to view a list of the available shortcut menus. In general, you will not need to explicitly set a control's shortcut menu. If your form calls the LIMSFormOpen procedure in its Open event, the procedure will automatically set the proper shortcut menu for text box and combo box controls.

Customizing Guidelines

Before beginning to customize J STREET LIMS, it is important to understand the features designed to support customizing and the version update process. Customizing within the guidelines described below will preserve your customizations while maintaining the ability to install J STREET LIMS version updates.

As described earlier, J STREET LIMS consists of LimsCode, the front-end database, and the SQL Server back-end database. LimsCode contains Access database objects: Tables, Forms, Queries, Reports, Macros, and Modules. The SQL Server back-end database contains tables and their associated indexes and relationships, views, and stored procedures. When a version update is provided, the installation software may replace any LimsCode object with an updated version, excluding those objects specifically provided for customizations. The version update may also include scripts to update the back-end database by adding columns to tables, adding indexes and relationships and adding new tables. Since a version update does not replace LimsCode but only those LimsCode objects that have changed or are new, sites can customize by adding their own LimsCode objects.

The simplest LimsCode customizations involve adding new objects. When adding a new object, choose a naming convention that will avoid conflicts with future J STREET LIMS objects and quickly identify the object as a site-specific addition. For example, ABC Company developers may choose to include a company identifier in new object names:

```
frmABCMyForm
rptABCMyReport
sysABCMyTable
```

The naming convention in the example above will also group all of the ABC Company additions together in the sorted object lists.

Beginning with J STREET LIMS v3, significant changes were incorporated in the base system to isolate customizations. For example, new subforms were added to the most often customized forms to allow adding custom fields and unique business rules. In the base system, these subforms have no controls but have all the *hooks* necessary to support site-specific fields and customizations. Since these subforms were specifically designed to support customizations, they are unlikely to be replaced in a version update and provide a location to install custom features (see Customizable Subforms on page 28 for more information). System reports can also be easily customized by installing replacement versions leaving the base system reports intact (see Customizing System Reports on page 38 for more information).

When an existing LimsCode object must be customized, care must be taken to preserve the original object as well as the customized version. This is accomplished by maintaining three copies of the object: the original unmodified but renamed object, the new customized object, and the *installed* object. For example, the developers at ABC Company have decided to customize the J STREET LIMS main menu. They begin by copying the original main menu form to a new form appending "JS" to its name to save an original copy (in version 5 and earlier the appending identification was "MSC"). Next, the developers make another copy appending "ABC" to its name and they update this object with the necessary customizations. Once tested, they install their updated menu by copying it and overwriting the original object. Their customized LimsCode now includes the following objects:

```
frmMainMenu          ' installed version
frmMainMenuABC       ' customized version
frmMainMenuJS        ' original unmodified version
```

Now that ABC Company's LimsCode includes customizations for existing J STREET LIMS objects, the developers must review the updated object list before installing any J STREET LIMS version update. No additional action is required if frmMainMenu is not included in the version update's object list. If frmMainMenu does exist in the update's object list, the developers must review the update's ***Release Notes*** to learn what changes have been made to the system's main menu. If the changes are minor and not required they can install the update then save the updated main menu to the "JS" named object then reinstall their "ABC" named version. If the changes are important the developers would have to reapply their customizations to the main menu and save the three versions of the object.

The remaining chapters of this guide provide additional information to consider when customizing J STREET LIMS. The chapters below are organized by Access database object type.

Chapter 3: Tables

This chapter outlines the types of database tables used in the system and provides a detailed description of the system's internal configuration tables.

SQL Server Tables and Views

The SQL Server back-end database is the repository for all user-entered data. In single-user configurations this database resides on the workstation along with LimsCode. In multi-user configurations, the database resides on a server. In either case LimsCode *attaches* the back-end tables. Tables attached with an ODBC driver are identified by an arrow and globe icon in the LimsCode table list in the Access database window.

The back-end database model includes many tables, indexes, relationships, views, stored procedures and other rules which enforce the model's integrity constraints. Use SQL Server Management Studio (SSMS) to explore the database model. Note that existing back-end table relationships should not be changed.

Simple back-end customizations include adding new columns to existing tables, changing column lengths, and adding table or column constraints.

Following is a list of the current back-end database tables and views. Note that views are named beginning with a lower case 'v'.

Analysis	Analysis definitions
AnalysisQCDataType	QC data types configured by analysis
AnalyticalBatch	Analytical batch identification
ContainerType	Container types
Customer	Customer information
CustomerContactHistory	Contact history records
CustomerCostAnalysis	Costs of analyses overridden by customer
CustomerProjectMessaging	Customer messaging options by project
CustomerSampleLogin	Customer sample login options
Employee	Employee records
ErrorLog	System error log
Frequency	Sample scheduling frequencies (not configurable through any LimsCode screen)
Instrument	Instrument records

Laboratory	Outside laboratory information
Location	Sampling locations
LoginBatch	Login batch headers
LoginBatchSample	Login batch sample definitions
MainMenu	This table defines the available commands on the main menu as well as the minimum security role required to run the command and the DoCmd parameters. Menu commands can set frmGlobalValue fields to specific values and can include conditions to evaluate before running the menu command. Open this table in design mode to view a description of each column. Site-specific customizations can be added to the Custom tab of the main menu by adding records to this table with the Page column set to six (sixth tab of the menu).
MessageQueue	Messaging's queue of messages to send
MessageStyle	Messaging's message styles
Method	Method definitions
MethodCertification	Employee method certifications
MiscellaneousList	Audit trail reasons, SQL expressions, label scripts, text lists, SysCfg records, etc.
OtherCertification	Employee "other" certifications
Preservative	Preservatives
Procedure	QC procedures list
Project	Project headers
ProjectAnalysis	Project analysis configuration
ProjectRequirement	Project requirement configuration
QCData	QC data for analytical batches
QCDataType	QC data type definitions
QCSampleAnalysis	Sample analyses for QC project samples
Report	User-defined report definitions
ReportColumn	User-defined report column definitions
Requirement	Requirements
RequirementAnalysis	Requirement analysis configuration
ResultType	Result type definitions
Sample	Sample headers (regular and QC)

SampleAnalysis	Sample analyses for regular (non-QC) project samples
SampleAuditTrail	Sample audit trail data
SampleQuerySelect	SampleIDs by form, user, and computer resulting from sample queries
Sampler	Sampler records
SampleStatus	Sample statuses
SampleType	Sample types
Schedule	Current sample schedules and schedule histories
SystemConfiguration	System configuration data (single record)
Training	Employee training records
Units	Units of measure
Version	Database version number (single record)
vSampleExceptionQC	Sample exception view on Sample, QCSampleAnalysis, and Analysis tables
vSampleExceptionQCData	Sample exception view on SampleAnalysis, QCData, and QCDataType tables
vSampleExceptionStandard	Sample exception view on Sample, SampleAnalysis, and Analysis tables
vSampleTracking	Sample tracking view
vSampleTrackingUnion	Sample tracking view on SampleAnalysis and QCSampleAnalysis tables

Configuration Tables

LimsCode includes a number of tables that are used to configure the front-end software. All such tables are named beginning with lower case “sys” to distinguish them from attached SQL Server database tables and to reflect their internal system nature.

See Customizing Guidelines on page 13 for more information.

When customizing any configuration table, the standard practice of maintaining original, customized, and installed copies of database objects should be followed to avoid loss during installation of a version update. Following is a description of LimsCode’s configuration tables.

See page 33 for more information on adding bench sheets.

sysBenchSheet This table defines the system’s installed bench sheets. At a minimum a record must include the bench sheet name and the name of the bench sheet’s report object. If the bench sheet uses a setup form, such as the standard frmBenchSheetSampleQuery, the form object name should be included in the Subform column. Use the Inactive column to prevent further use of a bench sheet.

sysExpression This table defines the list of predefined expressions available in the analyte comparison setup screen's report fields list, the expression pick list of the user-defined report's column configuration subform, and the label style field pick lists on the system configuration screen. To add a new expression, the ExpressionType must be set to "AnalyteComparison", "UDRColumn", or "LabelField" and the expression must be valid for the columns available in qryAnalyteComparisonRpt, qryReportData, or qryContainerLabel. Note that this table could also be used to configure expressions for site-specific customizations by defining a unique ExpressionType.

sysFormat This table defines the system's available date and time formats, user-defined report column formats, and result value report formats. The records with a FormatType of Date or Time define the format options available under the system configuration screen's time and date options. Note that only these records use the Format and optional InputMask columns. The records with a FormatType of UDRColumn define the available format options for UDR column definitions. For these records the FormatName field is the actual value used for the UDR columns Format property. Additional site-specific formats for Date, Time and UDRColumn can be added to this table as well as new FormatType's to support customizations.

See Global Values on page 9 for more information.

sysGlobalValues This table contains a single record used to define the LimsCode version number, required database version, as well as licensee information. Use this table to add main menu and report logos. Table columns are also available to add a company name to the main menu and to adjust the company name and logo size and position on the main menu. Note that the single record in this table and sysWorkstationConfiguration and the attached SystemConfiguration table are used by qryGlobalValues to provide the underlying data for frmGlobalValues. TIP: to adjust the company name and logo settings, Ctrl+right-click the company name or logo border on the main menu while logged on as a member of the Owners security role.

See Label Styles on page 34 for more information.	**sysLabelStyle**	Use this table to install new container label styles. These styles are used in the project setup screen to define a label style for project samples. Enter a Name, Description and identify the report object in the Report column. Use the SetupForm column to identify the name of the form object used to prompt for additional label setup information. If no SetupForm is used the standard label quantity prompt is used. Use the LabelScriptTokens field to specify any valid label script tokens other than style and quantity supported by this label style. Use the ReportSettings column to change the report's Printer object settings (see procedure ReportPageSetup in the LIMS Report Functions module for supported settings).
	sysReport	This table is used to install a custom version of a base system report. See Customizing System Reports on page 38 for more information.
See UDR Templates on page 35 for more information.	**sysReportTemplate**	This table lists the user-defined report templates currently installed. After creating a new UDR template by copying and modifying an existing template, add a record to this table to install the template.
	sysSort	This table defines the available sorting options and corresponding expression for control chart, UDR, sample summary, and work sheet reports. Note that any expressions added for these reports must be valid for the reports underlying query. Site-specific sorting options for customizations can be added to this table by selecting a new and uniqe SortType name.

Temporary Records Tables

All of the LimsCode tables with names beginning with "sys" that are not configuration tables (see previous section) are temporary data tables. These tables are identified by the text "Temporary Records" added to the table's Description property.

Temporary records tables are used for a number of purposes where the system needs temporary storage. For example, adding or editing samples in the Batch Login screen uses a temporary records table. Whenever a system process uses a temporary records table, all records are first deleted from the table. All records in all temporary records tables are deleted when exiting LimsCode if the workstation configuration screen's "Delete temporary records at exit" option is enabled. When adding a customization that requires a new temporary data table, remember to include "Temporary Records" in the table's Description property and add delete permissions to the Admin user so records can be deleted automatically.

Chapter 4: Queries

This chapter describes the programming conventions used to implement the system's queries.

Overview

J STREET LIMS makes extensive use of Access queries. These queries range from simple select queries to crosstab and union queries. Select queries are commonly used to retrieve, order, and filter data from one or more tables. Crosstab and union queries are used to generate system reports. Update and append queries are also used for specific purposes. Note that these queries may set the query's RunPermissions property to "Owner's" (i.e. "WITH OWNERACCESS OPTION" in SQL) to override the user's permissions.

See Customizing Guidelines on page 13 for more information.

Customizing existing queries is not recommended since any query may be replaced during installation of a version update. If an existing query must be customized, use the standard object copying and renaming strategy to maintain a saved original, customized, and installed version of the object.

TIP
Understanding the back-end database model is important when constructing custom queries. Many queries use inner joins when selecting data from multiple tables. However, inner joins can generate undesirable effects in some cases. For example, when joining the Sample table with the Customer table to show a sample's customer, an inner join will exclude samples that do not have a customer. Since the **CustomerID** column of the Sample table is optional, an outer join must be used.

When adding new custom queries to the system it is often quicker to begin by copying and modifying an existing qeury. This method may also avoid the inner vs. outer join problem described above.

Naming Conventions

All of the J STREET LIMS query objects are named beginning with lowercase "qry." Many queries are given the same name as the form or report in which they are used. Following are examples of forms and their record source queries:

```
frmSampleLoginAnalysis          qrySampleLoginAnalysis
frmSampleResultsAnalysis        qrySampleResultsAnalysis
```

```
frmSampleWarning                    qrySampleWarning
frmSetupProjectsAnalyses            qrySetupProjectsAnalysis
```

J STREET LIMS query objects also use a less formal naming convention to help identify the query's purpose. Following is a list of the more common query naming conventions.

qry…Rpt	These queries are used as the record source for a report.
qry…Select	These queries are based on a temporary data table and are used with record selection forms which are accessed from a report setup dialog's [Select] button.
qry…Selected	Similar to the qry…Select queries, these queries are based on a temporary data table but filter the results to include only user-selected records.
qry…By…	These queries identify the data sort order. For example, qryLocationByName lists the Location table sorted by Name.
qry…List	These queries are used as the record source for ComboBox controls (i.e. pick lists).
qry…Active…	These queries are used to exclude *inactive* records. For example, qryProjectActiveList is used as the record source for a project ComboBox control with only active projects listed.

Chapter 5: Forms

This chapter outlines programming conventions used to implement the system's data entry/inquiry forms and identifies all of the available subforms designed to support customizations.

LIMSFormOpen

All J STREET LIMS forms call the **LIMSFormOpen** procedure of the LIMS Form Processing module. Lightweight (i.e. no form module) forms use the expression "=LIMSFormOpen([Form])" in the form's open event property to call the procedure. Forms with a module call the procedure in their Form_Open event procedure.

Using a single **LIMSFormOpen** procedure allows all J STREET LIMS forms to inherit similar behavior and provides a single point where future changes or additions can be made for form open events.

Following is a list of tasks performed by **LIMSFormOpen**:

1. For each date and time control on the form, set the control's Format and InputMask properties using the system configuration settings (see System Time and Date Formats below). Date and time controls are identified by the following "token=value" syntax in their Tag property: ControlFormat=Date, ControlInputMask=Date, ControlFormat=Time, and ControlInputMask=Time.
2. For each text box and combo box control on the form, set the control's ShortcutMenuBar property (if not already set). Depending on the type of form and type of field, one of the following custom shortcut menus is used: LIMS Datasheet, LIMS Date, LIMS Time, LIMS TextBox, and LIMS ComboBox.
3. If LIMSFormOpen is called with the limsAEForm and/or limsCalledForm enumerated type values in its OpenOptions argument, the procedure calls AEFormOpen and/or CalledFormOpen.

System Time and Date Formats

J STREET LIMS' system configuration screen allows the LIMS administrator to select an appropriate time and date display format for data entry forms and system reports. So this configuration works system-wide, each form that includes time or date controls must set each control's Format and InputMask properties properly. Any custom forms should also follow this convention for a consistent user interface. J STREET LIMS includes functions ControlFormat and ControlInputMask to return the current system time/date format and input mask. The LIMSFormOpen procedure calls these functions to set the Format and Input mask property of any text box control. To ensure a custom form maintains the same behavior, make sure the form calls LIMSFormOpen in its open event and one or more of the following tokens exist in a date or time control's Tag property: ControlFormat=Date, ControlInputMask=Date, ControlFormat=Time, and ControlInputMask=Time.

Add/Edit Forms

An J STREET LIMS add/edit form (AE form) is used to add new database records and to edit existing records. When a database record contains many fields or a subform is required to add records to multiple tables in a one-to-many relationship, the continous form view (where one row represents one record) is inadequate. J STREET LIMS uses an AE form to accommodate these situations. An AE form allows the user to switch between add and edit form modes. The analysis setup form (frmSetupAnalyses) is an example of an AE form.

The quickest method to create a new AE form is to copy then modify an existing form. When an AE form must be created from scratch, follow the procedures below to correctly configure the form.

To create an add/edit form:

1. Create the form with the following properties:

```
Default View                Single Form
Views Allowed               Form
Scroll Bars                 Neither
Record Selectors            No
Navigation Buttons          No
Allow Filters               No
Allow Edits                 Yes
Allow Deletions             Yes
Allow Additions             Yes
Data Entry                  Yes
Cycle                       Current Record
Menu Bar                    LIMS Menu Default Form
```

2. Add the following controls to the form's header section:

lblAddMode	label control with Caption "ADD MODE"
lblEditMode	label control with Caption "EDIT MODE"
cbxSelectForEdit	combo box based on appropriate pick list

3. To initialize the form in add mode, hide cbxSelectForEdit, set the form's toolbar, and move to a new record, add the following line of code to the form's Form_Open event procedure:

```
LIMSFormOpen Me, limsAEForm
```

4. Insert the following text in the Tag property of the first control in the form's detail section:

```
    AEFormSetFocus=FirstControl
```

5. Add an AfterUpdate event procedue to the cbxSelectForEdit control to filter the detail section's record based on the ID in the combo box. For example:

```
Private Sub cbxSelectForEdit_AfterUpdate()

    ' Find record for item selected in cbxSelectForEdit combo box,
    ' enable controls in detail section, and go to first control
    ' in detail section.

    On Error GoTo cbxSelectForEdit_AfterUpdateErr
    Dim Tmp As Variant

    Tmp = EnableControls("Detail", True)
    DoCmd.ApplyFilter , "[AnalysisID] = _
       Forms![frmSetupAnalyses]![cbxSelectForEdit]"
    AEFormSetFocus ("FirstControl")
    Exit Sub

cbxSelectForEdit_AfterUpdateErr:
    ErrorLog Me.Name & ".cbxSelectForEdit_AfterUpdate", Error$
    Exit Sub

End Sub
```

6. Requery the cbxSelectForEdit control in the form's AfterDelConfirm event procedure:

```
Private Sub Form_AfterDelConfirm(Status As Integer)

    On Error GoTo Form_AfterDelConfirmErr
    cbxSelectForEdit.Requery
    Exit Sub

Form_AfterDelConfirmErr:
    ErrorLog Me.Name & ".Form_AfterDelConfirm", Error$
    Exit Sub

End Sub
```

7. When opening the form using the OpenForm method of the DoCmd object, set the data mode parameter to acFormAdd:

```
    DoCmd.OpenForm "frmSetupAnalyses", , , , acFormAdd
```

For more information on the operation of AE forms refer to the AEForm… procedures in the LIMS Form Processing module.

Calling/Called Forms

J STREET LIMS' calling/called form capability allows a form user to add new entries to a form's combo box by either double-clicking or right-clicking and choosing "Add to List" to open the combo box's corresponding setup dialog. The current form, or calling form, opens the called form. When focus returns to the calling form it checks to see if the called form added any new records and if so, updates the combo box to show the item for the last record added.

This calling/called form processing is accomplished using a combination of event procedures, module code, and global values controls. Any form or subform can easily be configured as a calling form. However, since creating a called form requires modifying the global values form and a module procedure, creating a called form is not a recommended customization. For completeness, the procedures required to create both form types are described below.

To call another form from a form or subform control:

1. In the calling control's Dblclick event procedure add a line to invoke the CallingFormCall procedure passing the name of the control's main form followed by the name of the form to call. This saves the main form's name in the called form's caller control in the global values form and opens the called form. For example:

   ```
   CallingFormCall Me, "frmNotebookMethods"    ' from main form
   CallingFormCall Me.Parent, "frmSetupAnalyses" ' from subform
   ```

2. In the calling control's MAIN form Activate event (not in the subform) see if we are regaining the focus from a called form and if we are then update the combo box control with the last item added by the called form. Then clear our form name from the called form's caller control in the global values form. For example:

   ```
   ' If we are coming back from a call to frmNotebookMethods
   ' update the combo box that called the form.
    If WhoCalled("frmNotebookMethods") = Me.Name Then
        LastAdded = LastAddedID("frmNotebookMethods")
        If Not IsNull(LastAdded) Then Me!cbxMethodID = LastAdded
        Me!cbxMethodID.Requery
        ClearWhoCalled ("frmNotebookMethods")
        Exit Sub
    End If
   ```

To create a called form:

1. Make sure the form is callable by adding two text controls to frmGlobalValues and adding the form (and its two controls) to the Select Case statement in GetGlobalControls procedure of the LIMS Form Processing module.
2. In the Form_Open event procedure add the following line to reset the "last ID added" control in the global values form to null:

   ```
   LIMSFormOpen Me, limsCalledForm
   ```

3. In the Form_AfterInsert event procedure save the ID of the last record added to the "last ID added" control of the global values form. For example:

   ```
   CalledFormAfterInsert Me, Me![LocationID]
   ```

4. In the Form_Close event procedure add the line below. If we were called by another form, this procedure returns the focus to the calling form (if it is still open).

   ```
   CalledFormClose Me
   ```

Sample Query Form

J STREET LIMS includes a collection of query-by-example controls that are used throughout the system wherever sample querying is supported. This capability is provided by a single form (frmSampleQuery) that is used as a subform in all forms that support sample querying. Any custom form can easily include sample qeurying features by using the capabilities of frmSampleQuery. Use the procedures below to add sample querying to a form.

1. Add and properly size a subform control in the form and set the following subform properties:

   ```
   Name             frmSampleQuery
   SourceObject     frmSampleQuery
   ```

2. Set frmSampleQuery's mode in the Form_Open event of the main form. The form can be used to query regular samples, QC samples, or both regular and QC samples. Setting the appropriate mode determines the contents and columns of the project, location, and sample type pick lists as well as the visibility of the regular vs. QC option group control. Use the appropriate global constant (gcREGULAR_AND_QC, gcREGULAR_ONLY, gcQC_ONLY) to set the form's mode then use the subform control's Requery method to apply the mode:

   ```
   Me!frmSampleQuery.Form!txtWhichSamples = gcREGULAR_AND_QC
   Me!frmSampleQuery.Requery   ' triggers form's Current event
   ```

3. To collapse any expanded multi-select list box when exiting the subform control, add an OnExit event procedure to the subform control:

   ```
   Private Sub frmSampleQuery_Exit(Cancel As Integer)

        On Error GoTo frmSampleQuery_ExitErr
        Me!frmSampleQuery.Form.OnSubformExit
        Exit Sub

    frmSampleQuery_ExitErr:
        ErrorLog Me.Name & ".frmSampleQuery_Exit", Error$
        Exit Sub

    End Sub
   ```

4. Use the subform's SampleQueryWhere property to retrieve a complete SQL WHERE clause constructed from the QBE controls. If the property value is a zero-length string, no QBE controls have been selected and the user receives an appropriate message. For example:

```
'  Build WHERE clause
sWhere = Me!frmSampleQuery.Form.SampleQueryWhere
If Len(sWhere) = 0 Then Exit Sub
```

Sample Select Form

All J STREET LIMS forms that use the sample query form (see above) also include a [Select] button to open the sample select form, which allows the user to refine their query by enabling and disabling individual samples. Any custom form that supports sample querying can also use the sample select form by adding a [Select] button and opening frmSampleSelect in the button's click event.:

```
DoCmd.OpenForm "frmSampleSelect", _
    OpenArgs:="CallingForm=" & Me.Name & _
              ";Caption=" & Me.Caption & _
              ";RecordSource=qrySampleSelect"
```

Note that frmSampleSelect expects three parameters in OpenArgs: CallingForm, Caption, and RecordSource. CallingForm must be set to the name of the form (i.e. Me.Name) and Caption to a suitable form caption for the sample select form. Set the RecordSource parameter in OpenArgs to the underlying record source that includes a Selected field along with all sample characteristics joined with the SampleQuerySelect back-end table. If the calling form has a text box control named txtSamplesSelected, frmSampleSelect's Form_Unload event will set the control to the number of samples selected.

Customizable Subforms

Beginning with J STREET LIMS v3, new subforms were added to many base system forms specifically to enable site-specific customizations. For example, if a new field is added to the back-end's Sample table, a new data entry control (e.g. text box, combo box, etc.) can be added to frmSampleLoginCustom so users can enter values for the new field during single sample login. Similarly, a new data entry control can be added to frmSampleBatchLoginCustom to enter field values during batch login.

Designed as placeholders for customizations, these subforms are unlikely to be replaced during a version update. In most cases, version updates can be applied without affecting existing customizations. In the base system, these subforms have no controls but they do contain all of the event procedures and other "hook" procedures required to support customization. All of the subforms were tested with a single custom field named "MyField" added to the parent form's underlying back-end table and a single text box control named "txtMyField" added to the subform. In the base system, the text boxes have been removed, however, you will find all of the original tested code (now commented out) in each subform, which demonstrates the typical requirements needed to add a custom field.

Note that Access' event model will trigger a subform's Open event before the parent form's Open event. This is an important feature that the customizable subforms can use to their advantage to alter properties of the parent form and its controls. For example, the subform's Open event procedure can alter the default value, control source and visibility of parent controls. Code in the subform's Open event can even change the source object of other subform controls on the parent form without customizing the parent form.

The table below shows all of the customizable subforms available in J STREET LIMS version 5.x.

Customizable Subform	Parent Form
frmExcelSampleImportCustom	**frmExcelSampleImport**
frmResultsByAnalyteCustom	**frmResultsByAnalyte**
frmSampleBatchLoginCustom	**frmSampleBatchLogin**
frmSampleBatchLoginSetupCustom	**frmSampleBatchLoginSetup**
frmSampleLoginCustom	**frmSampleLogin**
frmSampleQueryCustom	**frmSampleQuery**
frmSampleResultsCustom	**frmSampleResults**
frmSetupAnalysesCustom	**frmSetupAnalyses**
frmSetupCustomersCustom	**frmSetupCustomers**
frmSetupLoginBatchSampleCustom	**frmSetupLoginBatchSample**
frmSetupProjectsCustom	**frmSetupProjects**
frmSetupRequirementsCustom	**frmSetupRequirements**

To add a new field to one of the customizable subforms, first review the code behind the form following the example logic for the MyField field and txtMyField text box control. With the exception of frmSampleQueryCustom, follow the procedures below to add a custom field to a customizable subform:

1. Add the field to the back-end database table making sure to enable the Allow Nulls. Custom back-end table fields cannot disable Allow Nulls because entering the subform commits the parent form's record to the database, which could not occur if a required field was omitted. Validation to require a field value can be performed in the subform (see step 9 below).
2. Add unbound controls to the form and set the tab order.
3. Add "FirstControl" and "LastControl" to the Tag property of the first and last controls (could be the same control) in the subform's tab order.

4. Add module-level variables to support the parent form's Undo event.
5. Add code to clear each control and module-level variable in the form's ClearControls procedure.
6. Add code to restore each control's value from its module-level variable in the form's UndoControls procedure.
7. Add code to set each control's value and module-level variable from the parent table field in the form's UpdateControls procedure.
8. Add a BeforeUpdate event procedure for each control whose value must be saved to a back-end table field. Add code to the procedure to assign the control's value to the parent form's table field.
9. Add code to perform any field or form validation to the form's DataIsValid function. Note that required field validation for custom fields must be performed in DataIsValid and not by disabling the back-end table field's Allow Nulls property (see step 1 above for more information).

Chapter 6: Reports

This chapter outlines programming conventions used within the system's reports and the features available to install custom versions of system reports.

System Time and Date Formats

J STREET LIMS' system configuration screen allows the LIMS administrator to select an appropriate time and date display format for data entry forms and system reports. So this configuration works system-wide, each report that includes time or date controls must set each control's Format property properly. Any custom reports should also follow this convention for a consistent appearance.

See form time and date formats on page 23 for more information.

J STREET LIMS includes function ControlFormat to return the current system time/date format. While this function could be called in the Report_Open event to set the Format property for any time and date controls, this would prevent the use of lightweight reports (see below). To support the system time and date formats from lightweight reports, J STREET LIMS includes function ControlFormatOnReportOpen. This function will set the Format property to the system time/date format for any text box control where either the string "ControlFormat=Time" or "ControlFormat=Date" exists in the control's Tag property. ControlFormatOnReportOpen is automatically called by any of the generic report open handler functions listed in the next section. To ensure any custom report uses the system time/date format, simply enter the appropriate Tag string for each time/date text box and add the appropriate report open handler function to the report's OnOpen property.

Lightweight Reports

A lightweight object in Access is a form or report object whose HasModule property is set to No. This indicates the object does not have a class module (code behind the form or report). Creating report or form objects without a module improves the performance and decreases the size of the LimsCode database. Many of the LIMS system reports are lightweight reports. This is accomplished by using a common set of report controls and a set of generic event handler functions that are passed a reference to the report object. Calling the event handler is simply a matter of entering the expression =*GenericHandlerFunction*([Report]) in the property sheet of the report for the proper event.

To allow a custom report to display the standard logo and header and footer expressions defined in the system configuration screen, the report must use one of the report open handler functions (see below) and include the following controls:

fraLogo	Bound object frame for the logo
txtHeader1	Text box to display header line 1
txtHeader2	Text box to display header line 2
txtFooter2Left	Text box to display left-justified footer
txtFooter2Right	Text box to display right-justified footer

Following is a list of all the global event handler functions available in the LIMS Report Functions module.

Event Handler	Description
OnBenchSheetClose	Called from the OnClose property of bench sheet reports. Redisplays frmBenchSheetSetup if it is open.
OnBenchSheetOpen	Called from the OnOpen property of bench sheet reports. Sets menu bar, toolbar, logo, headers, footers, and control time/date format.
OnInvoiceClose	Called from the OnClose property of invoice reports. Redisplays frmInvoiceSetup if it is open.
OnInvoiceOpen	Called from the OnOpen property of invoice reports. Sets menu bar, toolbar, logo, headers, footers, and control time/date format. Sets Customer footer's ForceNewPage property using setup form’s chkForceNewPage control.
OnReportNoData	Called from the OnNoData property of any report. Displays a “Report has no data” message box.
OnReportOpen	Called from the OnOpen property of system reports. Sets menu bar, toolbar, logo, headers, footers, and control time/date format.
OnUDReportClose	Called from the OnClose property of UDR templates to redisplay frmReportGenerateSetup.

Event Handler	Description
OnUDReportOpen	Called from the OnOpen property of UDR templates. Calls fuction FormatReportTemplate of the LIMS Report Functions module to perform all formatting of the UDR's report template.

Bench Sheets

In its simplest form, a bench sheet contains nothing more than boilerplate text, lines, and rectangles and is not bound to a data source. More complex bench sheets can require custom setup forms, tables, queries, and custom code. Bench sheets that need a sample query dialog to select and display sample characteristics can use the standard bench sheet sample query form included in the system (frmBenchSheetSampleQuery). Add a record to LimsCode configuration table sysBenchSheet to install a new bench sheet. Following is a list of the columns in table sysBenchSheet.

BenchSheetID	AutoNumber primary key.
Name	Enter a unique name to appear in the bench sheet pick list.
Subform	Leave blank for a "boilerplate" bench sheet or enter "frmBenchSheetSampleQuery" for a bench sheet that uses the standard sample query dialog.
Report	Enter the name of the bench sheet report object.
Inactive	Set to 'No' so the bench sheet is included in the pick list.
Notes	Enter any notes or comments to describe this bench sheet (optional).

To create a simple "boilerplate" bench sheet:

1. Create a new report object by copying either the portrait or landscape version of the "LIMS Bench Sheet Template" report. Follow the standard bench sheet naming convention by beginning your bench sheet name with "rptBenchSheet".
2. Open the new bench sheet report in design mode and add the required text, lines, rectangles, etc. to the report's detail section.
3. "Install" the bench sheet by adding a record to the sysBenchSheet table leaving the Subform column blank.

To create a bench sheet that uses the standard sample query dialog:

1. The quickest method to create a new bench sheet is to copy an existing rptBenchSheet... object that uses the standard sample query dialog. Follow the standard bench sheet naming convention by beginning your bench sheet name with "rptBenchSheet".
2. Open the new bench sheet report in design mode and edit the layout accordingly. Note that this style bench sheet uses for its RecordSource qryBenchSheetSample which is populated by the standard bench sheet sample query form frmBenchSheetSampleQuery.
3. Copy any bound control and change its ControlSource property to display sample characteristics or use the View|Field List... menu entry and drag new fields to the report layout.
4. You can test this style bench sheet by opening the print preview window from the design window. However, the bench sheet will only display sample data if qryBenchSheetSample (actually table SampleQuerySelect) has data. To add sample data for testing, simply run any existing bench sheet that uses the standard sample query dialog. The samples queried for this style bench sheet are not overwritten until samples are queried for another bench sheet.
5. "Install" the bench sheet by adding a record to the sysBenchSheet table and entering "frmBenchSheetSampleQuery" in the Subform column.

Label Styles

To create a custom label style for sample container labels, begin by copying one of the existing rptContainerLabel… objects. If necessary, create a new qryContainerLabel… query for the new report's record source. Note that any new query should include a join with table sysSampleLabel. When J STREET LIMS generates labels, either interactively or automatically during batch login, a single record is written to table sysSampleLabel for each requested label. Multiple copies of the same label are produced by adding multiple sysSampleLabel records with the same sample ID. If necessary, create a custom label setup form to prompt the user for additional label data when printing interactively. For an example of this capability, refer to frmSampleContainerLabelSetup used for environmental style labels. When printing labels interactively, any label style that does not use a custom setup form automatically uses frmContainerLabelQuantity which simply prompts the user for label quantity.

NOTE

To prevent runtime warnings for unavailable printers, all container label report objects should be saved in design mode without a specific printer. The label printer is assigned at runtime using the label printer selections on the **Workstation Configuration** screen. To ensure a specific printer is not saved with the report object, open the report in design view, select the Page Setup ribbon command tab, click Page Setup, select the Default Printer option on the Page tab, click OK, then close and save the report.

Add a record to LimsCode configuration table sysLabelStyle to install a new label style. Following is a list of the columns in table sysLabelStyle.

BenchSheetID	Enter a unique number for the primary key.
Name	Enter a unique name to appear in the label style pick lists.
Description	Enter a description of the label's contents to appear in the label style pick lists.
SetupForm	Enter the custom setup form used when generating labels interactively.
Report	Enter the name of the label report object.
LabelScriptTokens	This column is used to specify a list of additional label script tokens (other than "Style" and "Quantity") for the style. These tokens are entered in label scripts which are used to print sample labels automatically during batch sample login. Currently, only ContainerType and Preservative are supported.
ReportSettings	Use this column to change the report's Printer object settings using *token=value* syntax (see procedure ReportPageSetup in the LIMS Report Functions module for supported settings).

UDR Templates

J STREET LIMS' user-defined reports (UDR) are formatted by modifying control properties of Access report objects at runtime. These report objects are the UDR templates associated with a UDR definition. J STREET LIMS includes a series of UDR templates and additional templates can easily be added.

To create a new UDR template, begin by copying an existing template. Copy rptTemplatePort for a portrait template, rptTemplateLand for a landscape template, or copy one of the existing templates that more closely resembles the desired template layout. A UDR template may include fixed columns to display sample characteristics as well as user-defined columns. The number of user-defined columns is identified in table sysReportTemplate (see below). Following is a list of the controls that must exist in the template. Note that control names ending in 'X' indicate duplicated controls for user-defined columns where 'X' is the column letter (e.g. A, B, C, etc.).

UDR Control	Description
txtReportTitle	Report title
fraLogo	Bound object frame for company logo
txtHeader1Left	Left-justified first header line
txtHeader1Right	Right-justified first header line
txtHeader2Left	Left-justified second header line
txtHeader2Right	Right justified second header line
lblLabelX	Column X label where X is the column letter (A, B, etc.). One control for each user-defined column.
lblUnitsX	Column X units. One control for each user-defined column.
txtValueX	Column X value. One control for each user user-defined column.
txtWarningX	Column X warning character. One control for each user-defined column. This column can be omitted if the WarningsColumn column of sysReportTemplate (see below) is set to No.
txtNotes	Bound to the Notes column of qryReportData displays the sample's notes.
txtConclusions	Bound to the Conclusions column of qryReportData displays the sample's conclusions.
txtFooter1Left	Left-justified first footer line
txtFooter1Right	Right-justified first footer line
txtFooter2Left	Left-justified second footer line
txtFooter2Right	Right-justified second footer line
txtMin	Set to "Minimum:" if statistic is enabled
txtMinX	The minimum of column X's values
txtMax	Set to "Maximum:" if statistic is enabled
txtMaxX	The maximum of column X's values
txtSum	Set to "Sum:" if statistic is enabled
txtSumX	The sum of column X's values

UDR Control	Description
txtAvg	Set to “Average:” if statistic is enabled
txtAvgX	The average of column X’s values
txtGeoMean	Set to “Geo. Mean:” if statistic is enabled
txtGeoMeanX	The geometric mean of column X’s values
txtStDev	Set to “Std. Dev.:” if statistic is enabled
txtStDevX	The standard deviation of column X’s values
txtReportText	Set to the report’s report text value

Although the UDR controls can be in any report section the logo, header, and column label and units controls are normally placed in the report’s page header section. The column value and warning controls and the notes and conclusions controls are placed in the detail section. The footer controls are placed in the page footer section. The summary statistics and report text control are placed in the report footer section. While the controls listed above must exist in the template, specific controls can be hidden by setting the control’s Visible property to ‘No’.

To install a new UDR template, add a record to table sysReportTemplate.

Add a record to LimsCode configuration table sysReportTemplate to install a new UDR template. Following is a list of the columns in table sysReportTemplate.

TemplateID	Enter a unique number for the primary key.
Name	Enter a unique descriptive name to appear in the UDR definition screen’s template pick list.
Report	The name of the Access report object.
MaxUserDefinedColumns	Enter the number of user-defined columns available in the template.
PaperSize	This column is no longer used in J STREET LIMS version 2.x. The paper size can be saved during template design using File|Page Setup. An Access 2.0 bug prevented the paper size from being saved with the report object.
WarningColumns	To gain additional column value space on a template, the txtWarningX columns can be omitted. Set this column to ‘No’ to let the system know the txtWarningX columns do not exist in the template.

NOTE

All default J STREET LIMS templates are configured to use the default Windows printer. To designate a specific printer for a UDR, create a new UDR template and use File|Page Setup from the Access report design window and select a specific printer. The printer information is saved with the Access report object definition. Use this technique when a specific UDR should always be sent to a designated printer.

Customizing System Reports

J STREET LIMS v6 includes an integrated mechanism that allows any system report to be replaced with a customized version. To customize a system report, copy the existing report to your own version using a suitable naming convention to identify your custom version (see Customizing Guidelines on page 13). Leave the original report unmodified and make all changes to your version of the report. So that your version of the report is used throughout the LIMS, add a new record to the sysReport table setting the Report field to the name of the base system object, and the CustomReport field to the name of your replacement version of the report.

Using the Sample Summary report as an example, here is how the replaceable report feature works. Throughout the system, wherever a system report is opened for printing or previewing, the system uses the following VBA code with the appropriate report object name:

```
DoCmd.OpenReport LIMSReport("rptSampleSummary")
```

The LIMSReport function, part of the LIMS Report Functions module, looks for a sysReport record for the passed report name. If no record exists the function simply returns the name of the report passed. If a sysReport record for the report is found, the function returns the report name listed in the CustomReport field. All system reports are opened using the LIMSReport function so any report can be replaced with a custom version by simply adding a record to the sysReport table. Note that sysReport is a LimsCode table so you must distribute an updated LimsCode to all LIMS workstations after adding your report and the sysReport record.

Chapter 7: Macros

This chapter outlines programming conventions used within the system's macros.

Overview

Since Access macros offer no error trapping mechanisms and they can be halted by users with Ctrl+Break, their use within J STREET LIMS is not recommended. LimsCode uses only two special purpose macros: AutoExec and AutoKeys.

See System Startup on page 10 for a description of the SystemStartup function.

The AutoExec macro is a special purpose macro. When Access opens a database, it looks for a macro with this name and, if it finds one, runs it automatically. The AutoExec macro in LimsCode uses a single RunCode action to run the SystemStartup function. Modifying the SystemStartup procedure is not recommended since it may be modified in future version updates. If site-specific startup tasks must be included, add a second RunCode action to the AutoExec macro to invoke a custom startup procedure added to a custom module.

The AutoKeys macro is another special purpose macro. Access allows keys or key combinations to be assigned to an action using the AutoKeys macro. J STREET LIMS includes two key assignments in the AutoKeys macro. Ctrl+D is assigned to a RunCode action to run the InsertCurrentDate function. Since the Access Ctrl+semicolon (;) keyboard shortcut to insert the current date in a field does not work with fields with an input mask, the InsertCurrentDate function and the Ctrl+D key combination was added to LimsCode to provide a workaround. Ctrl+T is assigned to a RunCode action to run the TextBuilder function which opens the Text Builder popup screen. Additional key combinations may be added to AutoKeys as necessary.

Chapter 8: Modules

This chapter outlines programming conventions used within the system's modules.

Overview

The J STREET LIMS modules included in LimsCode provide much of the system's VBA code. The rest of the system's VBA code exists as form and report class modules (i.e. code behind forms and reports). Modifications to the LimsCode modules are not recommended since any module may be replaced during installation of a version update.

Developers will find a wealth of VBA code resources within the LimsCode modules. Browsing the module's code and comments is a good method to uncover techniques to approach specific programming problems. While any LimsCode module may change with a version update, the module's procedure names and arguments should remain stable so they may be called from any custom procedures. Any custom procedures should be added to a custom module. When adding a custom module, use a naming convention that properly identifies the module and will not conflict with future modules.

The declarations section of each module includes a comment block with a module description and revision history. For reference, the following is a list of LimsCode modules that shipped with versions 5 and earlier:

Module	Description
LIMS Form Processing	Functions for add/edit forms, called/calling forms, etc.
LIMS Functions	Miscellaneous LIMS functions for system startup, parsing, version checking, etc.
LIMS Office Automation	Functions supporting integration with Microsoft Office applications including Excel interface functions.
LIMS Report Functions	System and user-defined report generating functions including generic event handlers for lightweight reports (see page 31 for more information on lightweight reports).
LIMS Sample Functions	LIMS sample-related functions including audit trailing, label printing, completing, assigning costs, etc.

Module	Description
MSC Functions	J STREET LIMS's miscellaneous functions including error logging, reattaching databases, verifying security roles, etc.

Version 6 Changes

As part of the migration from version 5 to J STREET LIMS version 6, the new development team introduced an extensive standard code library of VBA functions.

These battle-tested standard modules and class modules are used throughout several dozen other Access applications. As they are the byproduct of a different development team, they do not match the VBA naming conventions and coding style of the LIMS Modules listed above nor the "code-behind" modules of the LIMS forms and reports. Converting these code library modules to match the conventions from version 5 and earlier would break compatibility with the other Access applications that rely on them. Refer to https://nolongerset.com/building-your-library/ for additional context on this topic.

The newly added code library modules use the following naming conventions:

New Standard Modules by Prefix

[No prefix]	Generic code library routines used across many applications and grouped by functional area (e.g., FileFunctions, FormFunctions, StringFunctions, etc.)
jl	**J** STREET **L**IMS application-specific routines (e.g., jl_Main, jlFactory, jlRibbon, etc.)

New Class Modules by Prefix

[No prefix]	Singleton class modules with PredeclaredId Attribute set to True (e.g., Backend)
cls	Class modules used to encapsulate similar functionality (e.g., clsExcel)
coll	Strongly typed collection classes (e.g., collCmdBars)
ErrEx	vbWatchdog error handling classes (e.g., ErrExCallstack)
i	Interfaces (e.g., iMultiForm)
o	Discrete objects; these classes often have multiple instances (e.g., oRibbonNode)
we	Classes that primarily use the **WithEvents** keyword to handle form and control events generically (e.g., weResizer)

Error Handling via vbWatchdog

Rather than include boilerplate error-handling code in every procedure, the code library modules referenced above use a default global error handler as provided by the third-party add-in vbWatchdog. For additional information, refer to https://nolongerset.com/error-handling-evolution/.