Sage SalesLogix
LAN Developers Reference

Version 7.5

Developed by Sage SalesLogix User Assistance

sage software

Your business in mind.

Sage SalesLogix LAN Developers Reference

Documentation Comments

This documentation was developed by Sage SalesLogix User Assistance. For content revisions, questions, or comments, contact the writers at saleslogix.techpubs@sage.com.

Version

Version 7.5
091808

Technical Support

Technical Support is available to customers with support contracts directly from Sage Software and to Certified Business Partners. Calls are answered during business hours, Monday through Friday, excluding holidays. Current contact information is available on the Sage SalesLogix Web site. Customers with a valid technical support contract and a Web access code can request technical support electronically on the Sage SalesLogix SupportOnline/Sage Software Online Support and Services Web site.

Sage SalesLogix Web site: www.sagecrmsolutions.com
Sage SalesLogix SupportOnline/Sage Software Online Support and Services
Web site: http://support.saleslogix.com

Revisions to this book are posted on the Sage SalesLogix SupportOnline/Sage Software Online Support and Services Web site. Check this site regularly for current Sage SalesLogix product documentation.

Contents

Introduction

Welcome to the Sage SalesLogix LAN Developers Reference. This Developers Reference contains information about customizing Sage SalesLogix. Use the Architect to build and save your customized scripts and functions.

The Sage SalesLogix Professional Services Group or Sage SalesLogix Certified Business Partners can assist you with customizations, or special configuration and installation requirements.

About This Guide

The Sage SalesLogix LAN Developers Reference is available in two formats: PDF and compiled HTML (CHM). Both formats contain the same statements and functions. The PDF is provided for users who want to print the guide, either in sections or in full.

When copying sections of the sample code and pasting them into the Architect, use the HTML version of this guide for best results. If you copy from the PDF version, the integrity of formatting and spacing can be lost, causing Architect to generate an error message. This is an issue with PDF, not with the code samples provided.

What You Need to Know

The Sage SalesLogix LAN Developers Reference is written for software developers, Sage SalesLogix integrators, Business Partners, and information system (IS) professionals. This guide assumes that you have a working knowledge of the VBScripting Language, as well as ADO, and Relational Database Management Systems (RDBMS). The purpose of this guide is to provide information that will assist you as you customize Sage SalesLogix. This guide is not designed to teach you how to write code or script.

Typographic Conventions

The LAN Developers Reference uses the following typographic conventions.

Convention	Description
`monospace`	Indicates source code, structure syntax, examples, user input, and program output.
[ ]	Indicates optional syntax items. Type only the syntax within the brackets, not the brackets themselves.

Statement of Liability

Related Documentation

In addition to the Developers Reference, the following Sage SalesLogix documentation is available.

- The *Basic Language Reference* contains an alphabetic reference for objects, statements and methods in the Basic scripting language.
- The *Sage SalesLogix Planning Guide* provides installation and database recommendations, and guidelines to help you plan for a successful implementation. The *Sage SalesLogix Planning Guide* is available on Sage SalesLogix SupportOnline/Sage Software Online Support and Services at http://www.support.saleslogix.com.
- The *Sage SalesLogix Implementation Guide* provides step-by-step instructions for installing Sage SalesLogix in a Windows and Web environment. The *Sage SalesLogix Implementation Guide* is available on the Sage SalesLogix DVD.
- The *Sage SalesLogix Database* online Help documents all the tables and fields in the Sage SalesLogix base product as well as any optional bundles. Use this help file when you are adding custom tables and fields to the standard Sage SalesLogix database, or to build database queries.
- The *SalesLogix Legacy Basic LAN Developers Reference* provides Basic, COM, and SQL functions used in versions of Sage SalesLogix prior to 6.0. This document is provided to support backward compatibility and the use of these functions for new customizations is not recommended.

For additional information, refer to the Help systems within each application or the Sage SalesLogix SupportOnline/Sage Software Online Support and Services Web site: **http://support.saleslogix.com**

What’s New in the Sage SalesLogix Developers Reference

Activity List Main View API

Activity list main view objects, properties, and functions were exposed to allow you to:

- Hide and show tabs, modify their captions, and activate tabs.
- Remove, add, and enumerate time range filters and apply them.
- Enumerate, add, and remove calendar users.
- Change height of the preview pane and access any controls on it.

Refer to “ActivityListWindow Object” on page 130.

Activity Attachments

The Attachments property on the Activity object was exposed. Refer to“Activity Object” on page 124.

DataGrid Object: AllowNullBindID property

The AllowNullBindId property was added to the DataGrid object to allow the prevention of a datagrid query with a Null Bind Id on page load. Refer to “DataGrid Object” on page 134.

Activity User

Activity.User documentation was updated. Refer to “Activity Object” on page 124 and “User Object” on page 141.

ActivityDialog Object

The ActivityDialog Object was exposed to permit you to control the Activity Details dialog. Refer to “ActivityDialog Object” on page 128.

CalendarUsers Object

The CalendarUsers object was exposed. Refer to “CalendarUsers Object” on page 134.

Continued Support for Legacy Forms

Legacy Forms and Enable Basic scripts (legacy scripts) still function and can coexist with the new Forms and Scripts introduced with Sage SalesLogix v6.0 and later. You can still create legacy Forms and Scripts, however, doing so presents the following disadvantages:

- Most customizations consist of a View (user interface) and programming code or scripts that control the interface. In legacy, the View and script must work together but they are not linked in any way – it is up to the programmer to tie them together.
- Legacy Scripts are written in a Basic language called Cypress Enable. It is a standard Basic, similar to, but not exactly like Microsoft Visual Basic or VBScript. In Cypress Enable, all scripts are written in an editor similar to Notepad, which does not provide assistance entering or editing the code, nor help find errors or typographical mistakes.
- For information about legacy functionality, see the Sage SalesLogix Legacy LAN Developers Reference.

Functionality Added in Earlier Versions of Sage SalesLogix

In Sage SalesLogix versions 6.0 and later, customization code can be written in Microsoft VBScript. The script editor provides color-coding, ToolTips, and other features similar to Visual Basic. The editor supports the following:

Code completion. Code completion occurs when a control name and period are typed (“dot notation”). The applicable properties and functions appear in a list.

ToolTips. Function parameters display as a ToolTip after the function is typed.

Include Scripts. Global functions can be used across multiple scripts with the new “include scripts” option.

Find and Replace. Find and Replace capabilities allow you to search for specific words throughout the current script or all scripts in the database.

Chapter 1
Sage SalesLogix Functions and Objects

This section is an alphabetic reference for Sage SalesLogix objects, functions, and statements available in scripts and exposed through code completion. Each function lists syntax, parameters, returns (as applicable) and, in some cases, code samples. Some examples include references to related topics or other examples. Sage SalesLogix commands and functions are reserved words.

When copying sections of the sample code and pasting them into the Architect, use the HTML version of this guide for best results. If you copy from the PDF version, the integrity of formatting and spacing can be lost, causing Architect to generate an error message. This is an issue with PDF, not with the code samples provided.

The functions marked **-- Caution: Reserved --** are listed for your information, but should not be used. Sage SalesLogix makes no commitment to maintain these functions in future releases.

VBScript Functions

VBScript functions are available in Sage SalesLogix versions 6.0 and later. For information about VBScript, refer to the VBScript reference guide of your preference, or the Microsoft MSDN Web site.

Application Object Model

The Application object simplifies the task of dealing with Sage SalesLogix functions. The Application object is available to VBScript in the Sage SalesLogix Client without explicitly creating the object. Developers can use the Application object in VBScript by typing "Application" followed by a period and selecting the objects, methods, and properties from the drop-down list. Objects, methods and properties are listed here in the order visible in Architect, in the code completion drop-down lists.

For information on using the Application object model as a COM object, see Chapter 4, "Sage SalesLogix OLE DB Provider."

Finding Examples in the Sage SalesLogix Database

You can use the plugins in the Sage SalesLogix database to see how functions in this book are used in Sage SalesLogix.

To locate an example of using a function:

1. In **Architect**, on the **Edit** menu, click **Find**.
2. In the **Find Text** box, type the object name.
   For example, Application.BasicFunctions.ReportAddCondition.
3. In the **Search** group, select the **All Scripts** option.

4. Click **Find Next**.
   All search results are shown in a list at the bottom of the dialog box.
5. To open an item in the search results, select the item and then double-click.
   The plugin opens with the text you searched for highlighted.

Application.Activities

Add

Exposed In	Version 7.0
Function	Creates a new activity.
Object	Application.Activities.Add
Syntax	Add()
Parameters	Type - (Integer) a valid Activity Type.
	UserID - the Sage SalesLogix ID for the User.
Returns	Activity Object
Related Topics	N/A

You must use the Activity.Save method after creating a new activity.

GetActivityByID

Exposed In	Version 7.0
Function	Returns a single Activity object.
Object	Application.Activities.GetActivityByID
Syntax	GetActivityByID()
Parameters	ActivityID - a valid Activity Identifier.
Returns	Activity object
Related Topics	N/A

GetActivityList

Exposed In	Version 7.0
Function	Returns a list of Activity objects for a specific user and date range.
Object	Application.Activities.GetActivityList
Syntax	GetActivityList()
Parameters	UserID - the Sage SalesLogix ID for the User.
	StartDate - beginning date of a range.
	EndDate - end date of a range.
	IncludedUnconfirmed - (Boolean) If True, both confirmed and unconfirmed activities are returned. If False, only confirmed activies are returned.
Returns	List of Activity objects
Related Topics	N/A

Application.BasicFunctions

AddContactAndAccount

This function is deprecated in version 6.2.

Function	Launches the Add Account and Contact Wizard for the Legacy Basic Views in Sage SalesLogix. This function was replaced by Active Forms in release 6.2 and is supported for the convenience of customers who have been using Sage SalesLogix versions 6.1.x and earlier.
Object	Application.BasicFunctions.AddContactAndAccount
Syntax	AddContactAndAccount
Parameters	None
Returns	String
Related Topics	N/A

AddContactForAccount

This function is deprecated in version 6.2.

Function	Launches just the contact insert screen for a specific account for the Legacy Basic Views in Sage SalesLogix. This function was replaced by Active Forms in release 6.2 and is supported for the convenience of customers who have been using Sage SalesLogix versions 6.1.x and earlier.
Object	Application.BasicFunctions.AddContactForAccount
Syntax	AddContactForAccount ()
Parameters	AccountID as String - the Sage SalesLogix ID for the Account
Returns	Boolean
Related Topics	N/A

AddMinutesToDate

Exposed In	Version 6.2
Function	Adds minutes to a date, taking into consideration the office work week, as defined in the Administrator under System Information > Offices > Service/Support tab (double-click an office to access the Service/Support tab).
Object	Application.BasicFunctions.AddMinutesToDate
Syntax	AddMinutesToDate()

Parameters	OldDate as String	Start date
	Start as Long	Start time in minutes. For example, 7:00 a.m. = 420
	End as Long	End time in minutes
	Min as Long (minutes)	Minutes to add
	Offset as Boolean	If weekends are included in the work week, select True, otherwise set False.

Returns	String
Related Topics	N/A

AddPersonalContact

Function	Launches the Insert Contact screen with the seccode default value.
Object	Application.BasicFunctions.AddPersonalContact
Syntax	AddPersonalContact
Parameters	None
Returns	String
Related Topics	N/A

AllowPublicContactsOnPersonalActivities

Exposed In	Version 7.0
Function	Returns TRUE if the global option to allow non private contacts/accounts to be associated with the private activities is set to TRUE.
Object	Application.BasicFunctions.AllowPublicContactsOnPersonalActivities
Syntax	AllowPublicContactsOnPersonalActivities
Parameters	None
Returns	Boolean
Related Topics	N/A

AreAttachmentsSyncedToRemote

Function	Returns a Boolean based on whether or not the remote user has synchronization of attachments enabled by the administrator.
Object	Application.BasicFunctions.AreAttachmentsSyncedToRemote
Syntax	AreAttachmentsSyncedToRemote
Parameters	None
Returns	Boolean
Related Topics	N/A

CanAddPersonal

Exposed In	Version 5.2
Function	Returns True if the current user can add personal contacts.
Object	Application.BasicFunctions.CanAddPersonal
Syntax	CanAddPersonal
Parameters	None
Returns	Boolean
Related Topics	N/A

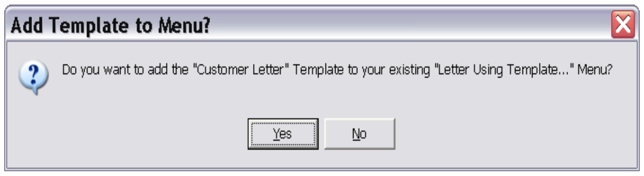
CascadeDelete

Exposed In	Version 6.2
Function	Allows the deletion of child records when the parent record is deleted and if the Cascade join type is set to Delete (in the Join Manager).
Object	Application.BasicFunctions.CascadeDelete
Syntax	CascadeDelete
Parameters	TableName - Base Table KeyFieldValue - Sage SalesLogix ID for the record being deleted.
Returns	Boolean
Related Topics	N/A
Example	When records are deleted from an ActiveX datagrid, the datagrid uses straight ADO to delete a record. ADO, however, does not adjust to accommodate the Sage SalesLogix database structure (Cascade etc). To address this, add a Delete popup menu explicitly to the grid popup menu rather than using a default implementation. The following example deletes a Contact record and all records from associated child tables where the cascade join type is Delete.

```
Application.BasicFunctions.CascadeDelete("Contact", "CA2EK0013402")
```

CheckWriteMRUMenu

Exposed In	Version 6.2
Function	Used to check and update the most recently used (MRU) menu items that display under the Write>E-mail, Fax, or Letter Using Template menus. If the menu item represented by the PluginID parameter has not been added to the menu that corresponds to the MergeMode parameter (E-mail, Fax, or Letter), the following dialog displays if the PluginID exists in the database and has not already been added to the corresponding menu. If the PluginID exists, CheckWriteMRUMenu returns True.

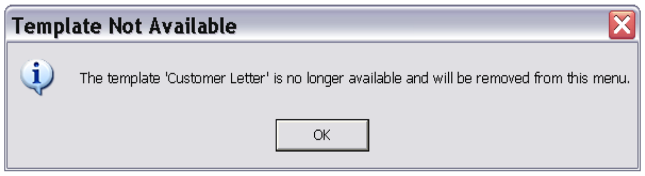


Add Template to Menu?

Do you want to add the "Customer Letter" Template to your existing "Letter Using Template..." Menu?

If the user clicks Yes the template is added to the corresponding MRU menu.

If the template represented by the PluginID does not exist in the database, but a corresponding menu item does exist, the following dialog displays. If the PluginID does not exist CheckWriteMRUMenu returns False:


Template Not Available

The template 'Customer Letter' is no longer available and will be removed from this menu.

Object	Application.BasicFunctions.CheckWriteMRUMenu	
Syntax	CheckWriteMRUMenu ()	
Parameters	PluginID	String. Represents a type 25 Mail Merge Template
	MergeMode	Long 0 = E-mail 1 = Fax 2 = Letter
Returns	Boolean	
Related Topics	N/A	
Example	See the example for "SelectTemplateEx" on page 56.	

CloseAllGroups

Function	Closes all Group Handle(s) currently open. The purpose of this function is to close a Group's handle to free the utilized memory. This function also closes any group handle opened by a nested script.
Object	Application.BasicFunctions.CloseAllGroups
Syntax	CloseAllGroups
Parameters	None
Returns	Boolean
Related Topics	"GetGroupIDs", "GetGroupValue", "GetGroupCount", "CloseGroup"

CloseCurrentView

This is a legacy function and should no longer be used, especially when invoking a view as a MainView.

Function **Bound** - Using the CloseCurrentView method on a bound form and passing in a True/False value results in the following:

True	Closes the view without posting any changes that may have incurred. (Basically setting the ModalResult of the form to mrCancel.)
False	Closes the view posting any changes that may have incurred. (Basically setting the ModalResult of the form to mrOK.)

Unbound - Using the CloseCurrentView method on an unbound form, the parameter is ignored and a value of True is always used since no binding has occurred.

Object	Application.BasicFunctions.CloseCurrentView
Syntax	CloseCurrentView(*Cancel*)
Parameters	Cancel as Boolean
Returns	Boolean
Related Topics	N/A

Example

```
If MsgBox("Would you like to save changes?", vbYesNo) = vbYes Then
Application.BasicFunctions.CloseCurrentView(False)
Else
Application.BasicFunctions.ClosecurrentView(True)
End If
```

CloseGroup

Function	Closes a Group's handle to free the memory utilized.
Object	Application.BasicFunctions.CloseGroup
Syntax	CloseGroup ()
Parameters	GroupHandle as String. The GroupHandle Name must be the name of a valid group handle opened by GetGroupIds.
Returns	Boolean
Related Topics	"GetGroupIDs", "GetGroupValue", "GetGroupCount", "CloseAllGroups"

ColorToString

Function	Given an RGB value, this function returns the named value the system uses to represent the color. Used with StringToColor.
Object	Application.BasicFunctions.ColorToString
Syntax	ColorToString (*Value*)
Parameters	Value as long - a numeric colorcode
Returns	String
Related Topics	"StringToColor"

For localization: the integer colorcode returns the *English* color string only.

CompleteActivity

Function	Displays the completion view for the given Activity.
Object	Application.BasicFunctions.CompleteActivity
Syntax	CompleteActivity()
Parameters	ActivityID (String) – a valid Activity Identifier

Returns	Boolean. If the Activity is complete, returns True. If the activity is not complete, returns False. If the Activity cannot be edited by the user, immediately returns False.
Related Topics	"ShowActivityNotePad", "EditActivity"

ComposeEMail

Function	Opens the e-mail editor and passes the address into the e-mail address To: field.
Object	Application.BasicFunctions.ComposeEMail
Syntax	ComposeEMail
Parameters	Address as String - the e-mail address for the addressee.
Returns	Boolean
Related Topics	N/A

ControlDo

Function	Allows the user to perform one of the following actions on a given control: Focus, BringToFront, SendToBack, Hide, Show Popup (can be used for controls with the ButtonVisible property).

This function is supported for compatibility reasons and can be used only if a new Active Script is set as a handler of an old object on a legacy view.

Object	Application.BasicFunctions.ControlDo
Syntax	ControlDo (*Object*, *Verb*)

Parameters	Object (name)	The name of the control.
	Verb	Allows the user to perform one of the following actions on a given control: Focus, BringToFront, SendToBack, Hide, Show, PopUp (can be used for controls with the ButtonVisible property)

Returns	String
Related Topics	N/A

ControlPrimaryVerb

Function	Given a specified control, returns its default or primary verb. Used with *ControlDo*. This function is supported for compatibility reasons and can be used only if a new Active Script is set as a handler of an old object on a legacy view.
Object	Application.BasicFunctions.ControlPrimaryVerb
Syntax	ControlPrimaryVerb (*aObject*)
Parameters	Object (name) as String
Returns	String
Related Topics	"ControlDo"

ControlVerbs

Function	Given a specified control, returns a string of the available verbs. Used with *ControlDo*. This function is supported for compatibility reasons and can be used only if a new Active Script is set as a handler of an old object on a legacy view.
Object	Application.BasicFunctions.ControlVerbs
Syntax	ControlVerbs (*aObject, Result*)
Parameters	Object (name) as String - The name of the control
Returns	String - List of the available parameters.
Related Topics	"ControlDo"

CopyAttachment

Exposed In	Version 6.2
Function	Copies the location of the attachment to the clipboard. A copy of the attachment can be pasted into Windows Explorer or to the Windows desktop.
Object	Application.BasicFunctions.CopyAttachment
Syntax	CopyAttachment()
Parameters	AttachID as String - KeyField ID for the Attachment table.
Returns	Boolean
Related Topics	N/A

CopyPlugin

Exposed In	Version 5.2
Function	This procedure takes a pluginID and makes a copy of the plugin. The parameters, PluginName, PluginDescription, and PluginFamily are the name, family, and description to use for the new plugin record. An error occurs if a plugin of the same name, family, and type already exists.
Object	Application.BasicFunctions.CopyPlugin
Syntax	CopyPlugin()
Parameters	SourcePluginID (String) - the ID for the plugin that is to be copied PluginName (String) - the name to use for the new plugin record PluginDescription (String) - the description to use for the new plugin record PluginFamily (String) - the family to use for the new plugin record
Returns	Boolean
Related Topics	N/A

CreateActivity

Function	Triggers the Create Activity dialog. You can supply default values for ContactID, OpportunityID, Regarding, Notes, and Leader.
Object	Application.BasicFunctions.CreateActivity
Syntax	CreateActivity ()

Parameters	Type as String - A Type is one of "Phone Call," "Meeting," or "To-Do"
	ContactID as String - the Sage SalesLogix ID for the contact.
	OpportunityID as String - the Sage SalesLogix ID for the opportunity.
	Regarding as String
	Notes as String
	Leader as String
Returns	String
Related Topics	N/A

CreateAdHocGroup

Exposed In	Version 6.2.1
Function	Creates an ad hoc group from a list of CRLF keys.
Object	Application.BasicFunctions.CreateAdHocGroup
Syntax	CreateAdHocGroup
Parameters	MainTable (String) – Name of the Main Table this group is based on.
	aGroupName (String) – Name of the group to be created.
	aIDs (String) – a CRLF delimited list of Keys from the Main Table.
	aStartingID (String) – First ID within the ID list to begin creation from. If empty, the complete ID list is used.
	aLayoutGroupID (String) - Specifies the ID of the group whose layout should be used when creating the group.
Returns	Boolean
Related Topics	N/A

CreateCompletedActivity

Function	The function triggers the Create Completed Activity dialog. You can supply default values for ContactID, OpportunityID, Regarding, Notes, and Leader. You can also pull "Regarding," "Notes," and "Leader" from the Sales Process table.
Object	Application.BasicFunctions.CompleteCreatedActivity
Syntax	CreateCompletedActivity*()*
Parameters	Type as String - A Type is one of "Phone Call," "Meeting," or "To-Do"
	ContactID as String - the Sage SalesLogix ID for the contact.
	OpportunityID as String - the Sage SalesLogix ID for the opportunity.
	Regarding as String
	Notes as String
	Leader as String
Returns	String
Related Topics	"CreateActivity"

CreateCompletedActivityEx

Exposed In	Version 6.2.1
Function	Triggers the Create Completed Activity dialog.
Object	Application.BasicFunctions.CreateCompletedActivityEx
Syntax	CreateCompletedActivityEx
Parameters	Type - string
	ContactID - string
	OpportunityID - string
	TicketID - string
	Regarding - string
	Notes - string
	Leader - string
Returns	Boolean
Related Topics	"CreateCompletedActivity"

CreateDocument

Function Triggers the Mail Merge dialog. You can supply default values for Entity, Template, Regarding, and other options. The Function returns True when the Merge button is clicked to merge the document, or False if the Cancel button is clicked.

Object Application.BasicFunctions.CreateDocument

Syntax CreateDocument()

Parameters All parameters are optional.

Type	String	Where to send the Mail Merge. Valid options are PRINTER, FAX, EMail and E-MAIL.
EntityID	String	UniqueID for the EntityIDType. Valid Options are Group PluginID or Family:Group Name (for Group), OpportunityID for OPPORTUNITY, ContactID for CONTACT.
EntityIDType	String	What to run the Mail Merge against. Valid Options are CONTACT, OPPORTUNITY, GROUP.
Template	String	The name of the Template to use in the mail merge.
Regarding	String	Data to populate the Regarding value for the mail merge.
RecordToHistory	Boolean	Whether to Record the mail merge to History or not. Valid Options are 1 (True), 0 (False).

Returns Boolean

Related Topics N/A

CreateLiteratureRequest

Function	Triggers the Create Literature Request dialog. You can supply default values for Description, Cover Letter file names, and Literature.
Object	Application.BasicFunctions.CreateLiteratureRequest
Syntax	CreateLiteratureRequest ()

Parameters	EntityID	String	UniqueID for the EntityIDType. Valid Options are Group Name (for Group), OpportunityID for OPPORTUNITY, ContactID for CONTACT.
	EntityIDType	String	What to run the literature request against. Valid Options are CONTACT, OPPORTUNITY, GROUP.
	Description	String	The name describing the literature request.
	Cover	String	The name of the Mail Merge template in the format Family:Name. The template must already exist in the database.
	Literature	String	The items of literature to be added to the request .

Returns	Boolean
Related Topics	N/A

CreateTempAdHocGroup

Function	Creates a temporary ad hoc group from a list of CRLF keys.
Object	Application.BasicFunctions.CreateTempAdHocGroup
Syntax	CreateTempAdHocGroup *MainTable, GroupName, IDs, StartingID*

Parameters	MainTable	String	Name of the Main Table this group is based on.
	GroupName	String	Name of the group to be created.
	IDs	String	A CRLF delimited list of Keys from the Main Table.
	StartingID	String	First ID within the ID list to begin creation from. If empty, the complete ID list is used.

Returns	Boolean
Related Topics	"SetCurrentClientGroup"

CreateTempContactGroupForAccount

Function	Creates a temporary group of Contacts for the Account. Also sets the current client group to this temporary group and makes the supplied ContactID the current record.
Object	Application.BasicFunctions.CreateTempContactGroupForAccount
Syntax	CreateTempContactGroupForAccount ()

Parameters	GroupName	String	The name of the group to be created.
	AccountID	String	The Sage SalesLogix ID of the Account.
	ContactID	String	The Sage SalesLogix ID of the Contact to set as the current record.

Returns	Boolean

Related Topics	"SetCurrentClientGroup"

CreateTempContactGroupForOpportunity

Function	Creates a temporary group of Contacts for an Opportunity. Also sets the current client group to this temporary group and makes the ContactID supplied the current record.
Object	Application.BasicFunctions.CreateTempContactGroupForOpportunity
Syntax	CreateTempContactGroupForOpportunity()

Parameters	GroupName	String	The name of the group to be created.
	OpportunityID	String	The Sage SalesLogix ID of the Opportunity.
	ContactID	String	The Sage SalesLogix ID for the Contact to set as the current Opportunity.

Returns	Boolean
Related Topics	"SetCurrentClientGroup"

CreateTempGroup

Function	Creates a temporary group from the Main Table with a single Where clause applied.
Object	Application.BasicFunctions.CreateTempGroup
Syntax	CreateTempGroup ()
Parameters	MainTable (String) – The name of the Main Table in the query that represents the group.
	GroupName(String)- The name of the group to be created.
	WhereField(String) – A Column in the query to be used in the Where clause to filter the Main Table result set.
	Op(String) – SQL operator for the Where Clause.
	WhereValue – Value to be used in the Where Clause.
	StartingID – Value of the first row in the result set to be used to create the group.
Returns	Boolean
Related Topics	"SetCurrentClientGroup"

CreateTempOpportunityGroupForAccount

Function	Creates a temporary group of Opportunities for an Account. Also sets the current client group to this temporary group and makes the OpportunityID supplied the current record.
Object	Application.BasicFunctions.CreateTempOpportunityGroupForAccount
Syntax	CreateTempOpportunityGroupForAccount()
Parameters	GroupName(String)- The name of the group to be created.
	AccountID(String) – The Sage SalesLogix ID of the Account.
	OpportunityID(String) – The Sage SalesLogix ID of the Opportunity to set as the current Opportunity.
Returns	Boolean
Related Topics	"SetCurrentClientGroup"

CreateTempOpportunityGroupForContact

Function	Creates a temporary group of Opportunities for a Contact. Also sets the current client group to this temporary group and makes the supplied OpportunityID the current record.
Object	Application.BasicFunctions.CreateTempOpportunityGroupForContact
Syntax	CreateTempOpportunityGroupForContact ()
Parameters	GroupName(String)- The name of the group to be created.
	ContactID(String) – the Sage SalesLogix ID of the Contact.
	OpportunityID(String) – the Sage SalesLogix ID of the Opportunity to set as the current Opportunity.
Returns	Boolean
Related Topics	N/A

CSVCount

Function	Takes a comma separated string and determines the number of items it contains.
Object	Application.BasicFunctions.CSVCount
Syntax	CSVCount (*aValue*)
Parameters	aValue {string} - a CSV string.
Returns	Integer - Returns the number of comma separated values in the specified string
Related Topics	"CSVField"
Example	The following example returns 2.

```
Application.BasicFunctions.CSVCount ("Phoenix, Arizona")
```

CSVField

Function	Returns the value from the comma separated string to which the "index" refers.
Object	Application.BasicFunctions.CSVField
Syntax	CSVField ()
Parameters	Value {string} - a CSV string.
	Index {integer} - where 0 represents the first item, 1 the second item and so on.
Returns	String
Related Topics	"CSVCount"
Example	

```
strData = Application.BasicFunctions.CSVField(strData,1)
strData = Application.BasicFunctions.CSVField(strData,2)
strData = Application.BasicFunctions.CSVField(strData,3)
```

CurrentAccountID

This function is deprecated in version 6.2.

Function	Returns the ID of the current account or returns a null string if there is no ID for the currrent account. This function was replaced by Active Forms in release 6.2 and is supported for the convenience of customers who have been using Sage SalesLogix versions 6.1.x and earlier.
Object	Application.BasicFunctions.CurrentAccountID
Syntax	CurrentAccountID
Parameters	None
Returns	String
Related Topics	"CurrentContactID", "CurrentOpportunityID"

CurrentAccountName

Function	Returns the name of the current account or a null string if there is no current account.

If you switch from the contact page to the account page using a link object, CurrentContactID returns the ContactID for the page you were on previously.

Object	Application.BasicFunctions.CurrentAccountName
Syntax	CurrentAccountName
Parameters	None
Returns	String
Related Topics	"CurrentContactID"

CurrentContactID

This function is deprecated in version 6.2.

Function	Returns the ID of the current contact or returns a null string if there is no ID for the current contact. This function was replaced by Active Forms in release 6.2 and is supported for the convenience of customers who have been using Sage SalesLogix versions 6.1.x and earlier.

If you switch from the contact page to the account page using a link object, CurrentContactID returns the ContactID for the contact page you were on previously.

Object	Application.BasicFunctions.CurrentContactID
Syntax	CurrentContactID
Parameters	None
Returns	String
Related Topics	N/A

CurrentGroupID

Function	Returns the PluginID of the current group.
Object	Application.BasicFunctions.CurrentGroupID
Syntax	CurrentGroupID
Parameters	None
Returns	String

CurrentOpportunityID

This function is deprecated in version 6.2.

Function	Returns the current Opportunity ID. If there is no current Opportunity ID, an empty string is returned. This function was replaced by Active Forms in release 6.2 and is supported for the convenience of customers who have been using Sage SalesLogix versions 6.1.x and earlier.
Object	Application.BasicFunctions.CurrentOpportunityID
Syntax	CurrentOpportunityID
Parameters	None
Returns	String
Related Topics	"CurrentAccountID", "CurrentContactID", "CurrentUserID"

CurrentUserID

Function	Returns the current user ID.
Object	Application.BasicFunctions.CurrentUserID
Syntax	CurrentUserID
Parameters	None
Returns	String
Related Topics	"CurrentAccountID", "CurrentContactID", "CurrentOpportunityID"

CurrentViewCancelShow

Function	Only effective during ONOPEN and ONCHANGE (or WHENOPEN and WHENCHANGE in legacy) for a displayed view (not a main view tab). This function stops the view from displaying, then returns 'Cancel' as the action taken on the view. The function is only effective before the view is actually shown.
Object	Application.BasicFunctions.CurrentViewCancelShow
Syntax	CurrentViewCancelShow
Parameters	None
Returns	None
Related Topics	"CloseCurrentView"

CurrentViewID

Function	Returns the ID for the active View. If the view is a managed view, then returns the ID of the row for the first grid found. If the view is currently adding a record instead of editing, the ID returned is blank (").
Object	Application.BasicFunctions.CurrentViewID
Syntax	CurrentViewID
Parameters	None
Returns	String
Related Topics	"CurrentAccountID", "CurrentContactID"

DateToISO

Exposed In	Version 6.2
Function	Converts a DateTime value to the ISO format (yyyymmdd hh:nn:ss).
Object	Application.BasicFunctions.DateToISO
Syntax	DateToISO()
Parameters	Date as Date
Returns	String
Related Topics	N/A

DeleteActivity

Exposed In	Version 6.2
Function	Removes an activity from the database.
Object	Application.BasicFunctions.DeleteActivity
Syntax	DeleteActivity()
Parameters	ActivityID as String.
Returns	Boolean
Related Topics	N/A

DeleteFileAttachment

Exposed In	Version 6.2
Function	Deletes the attached file reference from the database and launches a dialog box that provides the option to delete the file from the hard drive.
Object	Application.BasicFunctions.DeleteFileAttachment
Syntax	DeleteFileAttachment()
Parameters	AttachID as String - KeyField ID for the Attachment table.
Returns	Boolean
Related Topics	N/A

Dial

Function	Launches the System:SLX_CTI_CALL script which launches the Dialer.
Object	Application.BasicFunctions.Dial
Syntax	Dial()
Parameters	Number as String - home, work, or mobile phone (from Contact table).
Returns	Returns the phone number as Boolean, sets it to global, and calls System:SLX_CTI_CALL.
Related Topics	N/A

DoInvoke

Function	Allows access to the various plugin and function actions. For example, you can use this function to execute a process for printing a report.
Object	Application.BasicFunctions.DoInvoke
Syntax	DoInvoke (*Action, Argument*)

Parameters

Action(String) – defines the action to be invoked.

Argument(String) – defines the argument that is passed to the action.

Action	Argument
ActiveScript	System:Insert Ticket
Basic	System:SLX_Manage_AddItem
CrystalReportPreview	Account:Account Detail
CrystalReportPrint	Account:Account Detail
Execute	C:\WINNT\explorer.exe
Form	System:Add Edit Product
Function	Tools:Options
Lookup	Contact:Lastname
Macro	Personal:TestMacro1
MainView	System:Ticket Details
Open	C:\WINNT\explorer.exe
Print	C:\Program Files\SalesLogix\oainfo.ini
Process	Lead Track:Lead - Next Step
SQL	Personal:TestSQL1
View	Contact:Lead Sources
WordTemplateOpen	Letter:Base Letter
WordTemplatePrint	Letter:Base Letter

Returns None

Related Topics N/A

Example

```
Application.BasicFunctions.DoInvoke "Form", "Contact:Tickets"
```

When using the **Open** action to access a file whose name contains a comma, insert a forward slash before the comma. For example: Application.BasicFunctions.DoInvoke “Open”,”C:\te/,st.txt” opens file C:\te,st.txt.

EditActivity

Function Displays the edit view for the given Activity. If the Activity cannot be edited by the user, Cancel is immediately returned.

Object Application.BasicFunctions.EditActivity

Syntax EditActivity(*ActivityID*)

Parameters ActivityID (String) – a valid Activity Identifier.

Returns Boolean. Returns True if the OK button is clicked, returns False if the Cancel button is clicked.

Related Topics “ShowActivityNotePad”, “CompleteActivity”

EditEvent

Function	Displays the edit view for the given Event. This works only when the Event is for a user that is currently visible in the activity system, returns Cancel if the activity cannot be edited.
Object	Application.BasicFunctions.EditEvent
Syntax	EditEvent(*EventID*)
Parameters	EventID (String) – a valid Event Identifier
Returns	Boolean. Returns True if the OK button is clicked, returns False if the Cancel button is clicked.
Related Topics	“EditActivity”

EditFileAttachment

Exposed In	Version 6.2
Function	Opens the properties dialog box where you can edit the file name and file description.
Object	Application.BasicFunctions.EditFileAttachment
Syntax	EditFileAttachment(*AttachID*)
Parameters	AttachID as String - KeyField ID for the Attachment table.
Returns	Boolean
Related Topics	N/A

EditHistory

Exposed In	Version 5.2
Function	Displays the Edit History dialog for the HISTORY.HISTORYID.
Object	Application.BasicFunctions.EditHistory
Syntax	EditHistory (*HistoryID*)
Parameters	HistoryID
Returns	Boolean. Returns True if the OK button is clicked, returns False if the Cancel button is clicked.
Related Topics	“EditActivity”

ExportCurrentGroupToExcel

-- Caution: Reserved --

Function	Creates an Excel Spreadsheet based on the Current Group. **Two worksheets are created:** Sheet 1 is the Group List Sheet 2 is the Layout of the group list with format types
Object	Application.BasicFunctions.ExportCurrentGroupToExcel
Syntax	ExportCurrentGroupToExcel ()
Parameters	FileName (String) - the Excel file name to Save As if not displayed. ShowAfter (Boolean) - displays the Excel file after is it created.
Returns	None
Related Topics	N/A

Example (see SLX_Export_Group_To_Excel for full example)

```
oslxFuncs.ExportCurrentGroupToExcel strFileName, blShowAfter
If ErrorCheck ("Error exporting group in ExportCurrentGroupToExcel function:") > 0 Then
Exit Sub
```

ExportCurrentToExcel

Exposed In	Version 6.2
Function	Creates an Excel Spreadsheet based on records in the ListView.
Object	Application.BasicFunctions.ExportCurrentToExcel
Syntax	ExportCurrentToExcel ()
Parameters	File Name (String) - the Excel file name to Save As
	ShowAfter (Boolean) - displays the Excel file after is it created.
Returns	Boolean
Related Topics	N/A

FindNewOwner

Function	Allows the user to select a new owner for a record if they have ownership rights to the record.
Object	Application.BasicFunctions.FindNewOwner
Syntax	FindNewOwner ()
Parameters	OldID as String (SecCodeId)
	Maintable as StringSecCodeID (table name)
Returns	Boolean
Related Topics	N/A

GetActiveControlText

Exposed In	Version 6.2.1
Function	Retrieves the text of the active control (such as an Edit control or Memo control).
Object	Application.BasicFunctions.GetActiveControlText
Syntax	GetActiveControlText
Parameters	None
Returns	Boolean
Related Topics	N/A

GetApplicationPath

Function	Gets the full path to the Sage SalesLogix executable.
Object	Application.BasicFunctions.GetApplicationPath
Syntax	GetApplicationPath
Parameters	None
Returns	String. The full path to the the location where the Sage SalesLogix executable is installed.
Related Topics	N/A

Example

```
Sub Main
Dim strPath
StrPath = Application.BasicFunctions.GetApplicationPath
MsgBox StrPath
End Sub
```

GetAttachmentPath

Function	The file path to Sage SalesLogix attachments (as defined in the Administrator under System Information > Offices > Sync Options tab (double-click an office to access the Sync Options tab).
Object	Application.BasicFunctions.GetAttachmentPath
Syntax	GetAttachmentPath
Parameters	None
Returns	String file path to Sage SalesLogix attachments.
Related Topics	N/A

GetCrystalReport

Exposed In	Version 6.2.1
Function	Gets the named Crystal Report.
Object	Application.BasicFunctions.GetCrystalReport
Syntax	GetCrystalReport()
Parameters	ReportName - string
Returns	Object
Related Topics	N/A

GetDataPathValue

Function	Returns the value from the current view that has a datapath.
Object	Application.BasicFunctions.GetDataPathValue
Syntax	GetDataPathValue()
Parameters	DataPath {string}
Returns	String
Related Topics	N/A

GetDefaultAreaCode

Function	Returns the default area code.
Object	Application.BasicFunctions.GetDefaultAreaCode
Syntax	GetDefaultAreaCode
Parameters	None
Returns	String
Related Topics	N/A

GetDefaultSecCodeID

Function	Returns default security code ID.
Object	Application.BasicFunctions.GetDefaultSecCodeID
Syntax	GetDefaultSecCodeID
Parameters	None
Returns	String
Related Topics	N/A

GetDefaultWordProcessor

Exposed In	Version 5.2
Function	Gets the installed default word processor.
Object	Application.BasicFunctions.GetDefaultWordProcessor
Syntax	GetDefaultWordProcessor as String
Parameters	None
Returns	Returns "MSWord" or "Not Available".
Related Topics	N/A

GetDelimitedTerm

Function	Returns an item from a delimited list. This function, in conjunction with SetDelimitedTerm is useful for manipulating multiple values in a single string variable.
Object	Application.BasicFunctions.GetDelimitedTerm
Syntax	GetDelimitedTerm(*Value, Index, Delimiter)*
Parameters	Value (String) – delimited string. The delimiter can be any single character and, if left blank, is "|".
	Index (Integer) – position of item within the delimited string
	Delimiter (String) – character used to delimit string
Returns	String – An item within a delimited string
Related Topics	"GetDelimitedTermCount", "SetDelimitedTerm"

GetDelimitedTermCount

Function	Returns the number of items in a delimited list. The delimiter can be any single character and, if left blank, is "|".
Object	Application.BasicFunctions.GetDelimitedTermCount
Syntax	GetDelimitedTermCount(*Value, Delimiter)*
Parameters	Value (String) – delimited string
	Delimiter (String) – character used to delimit string

Returns	Integer – Number of items within a delimited string
Related Topics	"GetDelimitedTerm", "SetDelimitedTerm"

GetEMailType

Function	Returns "MSOUTLOOK", "SLMAIL" or "NONE".
Object	Application.BasicFunctions.GetEMailType
Syntax	GetEMailType
Parameters	None
Returns	String
Related Topics	N/A

GetGroupCount

Function	When used in conjunction with GetGroupValue, GetGroupIDs, CloseGroup, CloseAllGroups, this function returns the number of records in the Group.
Object	Application.BasicFunctions.GetGroupCount
SyntaxGetGroupCount(*"GroupHandle Name"[string]*)	**Parameters** GroupHandle as String - GroupHandle name must be the name of a valid group handle opened by GetGroupIDs.
Returns	A Long Value
Related Topics	"GetGroupIDs","GetGroupValue", "CloseGroup", "CloseAllGroups"

GetGroupIDs

Function	When used in conjunction with GetGroupValue, GetGroupCount, CloseGroup, CloseAllGroups, this function returns a reference to the Group record set.
Object	Application.BasicFunctions.GetGroupIDs
Syntax	GetGroupIds()
Parameters	Family Name as String. Group Name as String. GroupName can be the name of any "valid" group in Sage SalesLogix.
Returns	String
Related Topics	"GetGroupValue", "GetGroupCount", "CloseGroup", "CloseAllGroups"

Example

```
Sub ShowGroupIds
Dim strGroupID
Dim intCount
'Returns the unique IDs for the specified GroupName
strGroupID = Application.BasicFunctions.GetGroupIDs("Contact", "All Contacts")
For intCount = 1 to 5
  MsgBox "The PrimaryID for the current 'group' record is: " _
  & Application.BasicFunctions.GetGroupValue(strGroupID, intCount)
Next
End Sub
```

GetGroupList

Exposed In	Version 5.2	
	Returns a comma delimited, localized list of available groups.	
Object	Application.BasicFunctions.GetGroupList	
Syntax	GetGroupList () as String	
Parameters		
	Family as Variant	0 = Contact Groups
		1 = Account Groups
		2 = Opportunity Groups
		3 = Contract Groups
		4 = Defect Groups
		6 = Product Groups
		7 = Return Groups
		8 = Ticket Groups
		9 = TicketProblemType Groups
		10= TicketSolutionType Groups
		You may pass either the integer value defined above or the actual Family name (Account, Contact, Return, Lead, Campaign, custom table, and others.)
		This function can be used with any out of the box or custom group.
	Type as Integer	0 = Return Names
		1 = Return PLUGIN.PLUGINIDs.
Returns	String	
Related Topics	N/A	

GetGroupSQL

Exposed In	Version 5.2
Function	Returns the SQL statement used to generate the group.
Object	Application.BasicFunctions.GetGroupSQL
Syntax	GetGroupSQL ()
Parameters	ID as String - ID is the PLUGIN.PLUGINID for the group.
Returns	String
Related Topics	N/A

GetGroupValue

Function	Returns a string "handle" of all the unique IDs within a Sage SalesLogix group without having to load the group.
	GroupName can be the name of any "valid" group in Sage SalesLogix.
Object	Application.BasicFunctions.GetGroupValue
Syntax	GetGroupValue()
Parameters	GroupHandle as String
	GroupIndex as Long

Returns	String
Related Topics	"GetGroupIDs", "GetGroupCount", "CloseGroup", "CloseAllGroups"
Example	See"GetGroupIDs" on page 27.

The group member list is zero based. All *for loops* must start at 0 and loop until GetGroupCount(groupid) - 1.

GetIDFor

Function	Gets the new ID for a given table. If you are creating a new record for a table and need the ID before posting the record to the database, call this function to get a unique key to assign to your record. An example of a time when you might need to know the ID is when you are creating one or more records in a related table and need to populate the foreign key in that record and post the records together.
Object	Application.BasicFunctions.GetIDFor
Syntax	GetIDFor (TableName)
Parameters	TableName - the name of the table you are creating a record for.
Returns	New ID or Primary Key for a new record in the table.
Related Topics	N/A

GetLastMailMergeErrorMessage

Function	Returns a string of the last mail merge error that occurred from a call to either MergeFromFile() or MergeFromPlugin().
Object	Application.BasicFunctions.GetLastMailMergeErrorMessage
Syntax	GetLastMailMergeErrorMessage
Parameters	None
Returns	String
Related Topics	N/A

GetLastMailMergeErrorType

Function	Returns a long integer representing the last mail merge error type that occurred from a call to either MergeFromFile() or MergeFromPlugin(). Valid error types are as follows: Const errSuccess = -1 ' No error was reported by the mail merge engine. Const errAccessViolation = 0 ' An access violation occurred. Const errAttachmentPath = 1 ' The attachment path was not defined. Const errConnectionString = 2 ' The ConnectionString property was not set. Const errEmailSystem = 3 ' The EmailSystem property was not set. Const errException = 4 ' An internal exception occurred (generic). Const errHTTP = 5 ' A HTTP related error occurred (Web only). Const errInternal = 6 ' An internal error occurred. Const errInvalidEmail = 7 ' The e-mail address was invalid. Const errInvalidFax = 8 ' The fax number was invalid. Const errLibraryPath = 9 ' The library path was not defined. Const errMissingEmail = 10 ' The e-mail address was missing.

Const errMissingFax = 11 ' The fax number was missing.

Const errDefaultPrinter = 12 ' Not used.

Const errOther = 13 ' A known but undefined error occurred.

Const errOutlook = 14 ' A Microsoft Outlook related error occurred.

Const errQueryEmpty = 15 ' A query returned an unexpected empty result set.

Const errRemote = 16 ' The BaseKeyCode property was set and the Remote property was not.

Const errSiteCode = 17 ' The SiteCode property was not set.

Const errSLXDocument = 18 ' The Sage SalesLogix Document Description (*.sdd) could not be opened, created, or was invalid.

Const errTemplateID = 19 ' The requested TemplateID (type 25 template) was not found in the PLUGIN table.

Const errportType = 20 ' The TransportType property was not set.

Const errUnknown = 21 ' An unknown error occurred.

Const errUserID = 22 ' The UserID property was not set.

Const errWinFax = 23 ' A WinFax related error occurred.

Const errWord = 24 ' A Microsoft Word related error occurred.

Const errAbort = 25 ' The merge was canceled by the user.

Const errMergeSilently = 26 ' There was a conflicting property when MergeSilently was set to True.

Object	Application.BasicFunctions.GetLastMailMergeErrorType
Syntax	GetLastMailMergeErrorType
Parameters	None
Returns	Integer
Related Topics	N/A

Example

```
Dim vFileName
Dim vEngineErrorMessage
Dim vEngineErrorType
vFileName = "C:\Example.sdd"
If Not Application.BasicFunctions.MergeFromFile(vFileName) Then
vEngineErrorMessage = Application.BasicFunctions.GetLastMailMergeErrorMessage
vEngineErrorType = Application.BasicFunctions.GetLastMailMergeErrorType
If vEngineErrorType <> errAbort Then
  MsgBox "The following error was reported by the Mail Merge Engine: " & vEngineErrorMessage
End If
End If
```

GetLineCount

Function	Returns the number of lines of text in String.
Object	Application.BasicFunctions.GetLineCount
Syntax	GetLineCount ()
Parameters	aValue {string}
Returns	String
Related Topics	N/A

GetMenuSecurity

Function	Determines if the menu is available to the user (if it has not been restricted in menu security).
Object	Application.BasicFunctions.GetMenuSecurity
Syntax	GetMenuSecurity *(ID)*
Parameters	ID - accepts the index or the name of the Secured Function (from the SecFunction table).
Returns	True for Enabled / False for Disabled
Related Topics	N/A
Example	(see AccountDetail.SetAddress for complete example)

```
'Is Manage Alternate Addresses available, via Menu Security?
mnuAddress.Items.Items(intIndex + 1).Enabled = objBASIC.GetMenuSecurity(cViewAltAddress)
```

GetNthLine

Function	Given a block of text, this function returns the value of a specific line.
Object	Application.BasicFunctions.GetNthLine
Syntax	GetNthLine (*Value, Index*)
Parameters	Textstring as string Position as integer
Returns	Returns the Nth line, starting at 0 for String. A string with 5 lines has data returned for 0...4. Number out of range returns a blank string.
Related Topics	N/A
Example	In the following text: Sage SalesLogix 6.2 Test Application.BasicFunctions.GetNthLine(1) returns "6.2".

GetOwnerName

Function	Given a SecCodeID, this function returns the name of the owner, for example, Lee, Asia Pacific, and so on.
Object	Application.BasicFunctions.GetOwnerName
Syntax	GetOwnerName ()
Parameters	ID as String - equals the Sage SalesLogix ID.
Returns	String. Returns the ID of owner. If the owner is the user, the function returns the user name, otherwise it returns the owner name.
Related Topics	N/A

GetPersonalDataPath

Function	Gives the path to the user's local data.
Object	Application.BasicFunctions.GetPersonalDatapath
Syntax	GetPersonalDataPath
Parameters	None
Returns	The string path to the user's local data. If this path is unavailable, the application install path is returned.
Related Topics	N/A
Example	For a complete example, see the Sage SalesLogix VBScript System:SLX_Export_Group_To_Excel.

```
strPath = oslxFuncs.GetPersonalDataPath
strFileName = strPath & "\grp-"& strGroupName & ".XLT"
```

GetPluginListQuery

Exposed In	Version 5.2
Function	This function is a handle to a query with an open result set of plugins that match the criteria.
Object	Application.BasicFunctions.GetPluginListQuery
Syntax	GetPluginListQuery ()
Parameters	

PluginType	Integer	The TYPE field in the PLUGIN table to match on. (This function only returns a plugin list for one type at a time).
Family	String	The FAMILY field in the PLUGIN table to match on. If this is set to an empty string, plugins for all families is returned.
ShowPub	Boolean	Set to return public plugins (sometimes called system plugins)
ShowPri	Boolean	Set to return private plugins (sometimes called personal plugins)

Returns	A String representing a query.
Related Topics	N/A

GetPluginText

Exposed In	Version 7.0
Function	Calls the generic text plugin type.
Object	Application.BasicFunctions.GetPluginText
Syntax	GetPluginText (*FamilyAndName*)
Parameters	

FamilyAndName	Family and name of the text plugin.

Returns	String
Related Topics	N/A
Example	

```
Application.BasicFunctions.GetPluginText(“Personal:Test1”)
```

GetPrettyKeyPrefix

Exposed In	Version 6.2
Function	Returns the AlternateKeyPrefix for the Ticket. The AlternateKeyPrefix combined with the AlternateKeySuffix creates the ticket number (stored in the calculated field). The AlternateKeyPrefix is generated by an algorithm.
Object	Application.BasicFunctions.GetPrettyKeyPrefix
Syntax	GetPrettyKeyPrefix()
Parameters	Key as String
Returns	String representing the Alternate Key Prefix for the TicketID.
Related Topics	N/A

GetPrettyKeySuffix

Exposed In	Version 6.2
Function	Returns the AlternateKeySuffix for the Ticket. The AlternateKeyPrefix combined with the AlternateKeySuffix creates the ticket number (stored in the calculated field).
Object	Application.BasicFunctions.GetPrettyKeySuffix
Syntax	GetPrettyKeySuffix()
Parameters	Key as String
Returns	String, Bloolean
Related Topics	N/A

GetPrinters

Exposed In	Version 5.2
Function	Returns a comma delimited list of the available printers.
Object	Application.BasicFunctions.GetPrinters
Syntax	GetPrinters
Parameters	None
Returns	String
Related Topics	N/A

Example

```
vPrinters = GetPrinters()
```

GetTabVisibleProperty

Function	Allows a script to check the visibity status of a tab. For example, you might use this function to determine if tabs are visible or not visible.
Object	Application.BasicFunctions.GetTabVisibleProperty
Syntax	GetTabVisibleProperty (*Name)*
Parameters	Name. String value detailing the tab for which visibility status is being checked. The Name can be either the plugin name (in the Family:Plugin Name form) or the caption used to name the form.
Returns	Boolean
Related Topics	N/A

Example	The following example returns True if the Tickets tab is visible, False if the Tickets tab is not visible.

```
Application.BasicFunctions.GetTabVisibleProperty("Tickets")
Application.BasicFunctions.GetTabVisibleProperty("Account:Tickets")
```

GetTimeStampString

Function	Returns a DateTime stamp string.
Object	Application.BasicFunctions.GetTimeStampString
Syntax	GetTimeStampString
Parameters	None
Returns	String
Related Topics	N/A

GlobalInfoClear

Function	This method is used to clear the value of a global identifier in the Client. Uses the value of the variable to clear the value of a global identifier.
Object	Application.BasicFunctions.GlobalInfoClear Also see "Application.GlobalInfo Object" on page 87.
Syntax	GlobalInfoClear()
Parameters	ID as String – Identifier of the global variable
Returns	Boolean
Related Topics	"GlobalInfoSet", "GlobalInfoClear", "GlobalInfoExists", "GlobalInfoFor"

GlobalInfoClearAll

Function	This method is used to clear the value of all global identifiers in the Client.
Object	Application.BasicFunctions.GlobalInfoClearAll
Syntax	GlobalInfoClearAll
Parameters	None
Returns	None
Related Topics	N/A

GlobalInfoExists

Function	Used to determine if a global identifier is in the Client. Note that with this method, an identifier that has been cleared returns False. Also see "Application.GlobalInfo Object" on page 87.
Object	Application.BasicFunctions.GlobalInfoExists, or see Application.GlobalInfo.Object
Syntax	GlobalInfoExists()
Parameters	ID as String – Identifier of the global variable
Returns	Boolean
Related Topics	"GlobalInfoSet", "GlobalInfoClear", "GlobalInfoFor"

GlobalInfoFor

Function	This method is used to obtain the value of a previously set global identifier. Sets the variable to the value of a existing global identifier. Also see "Application.GlobalInfo Object" on page 87.
Object	Application.BasicFunctions.GlobalInfoFor (), or see Application.GlobalInfo.Object
Syntax	GlobalInfoFor()
Parameters	ID as String – Identifier of the global variable
Returns	String – Return value of the Identifier.
Related Topics	"GlobalInfoSet", "GlobalInfoClear", "GlobalInfoExists"

Example

```
strTest=Application.BasicFunctions.GlobalInfoFor("TestVar1")
```

GlobalInfoSet

Function	This method is used to set the value of a global identifier. Uses the value of the variable to set the value of a global identifier. Also see "Application.GlobalInfo Object" on page 87.
Object	Application.BasicFunctions.GlobalInfoSet, or see Application.GlobalInfo.Object
Syntax	GlobalInfoSet ()
Parameters	ID as String – Identifier of the global variable Value as String – Value to set.
Returns	Boolean
Related Topics	"GlobalInfoClear", "GlobalInfoExists", "GlobalInfoFor"

GroupQueryBuilder

Exposed In	Version 7.0
Function	Displays the QueryBuilder with the joins based on the specified table.
Object	Application.BasicFunctions.GroupQueryBuilder
Syntax	GroupQueryBuilder()

Parameters

MainTableName	Optional string containing the table name Query Builder uses to create join data. If value is empty the maintable from the Active main view is used. The default is Contact.
GroupName	Optional string containing default group name. The default is New Query.
TemplateGroupID	Optional string containing existing GroupID on which the new group is based (including properties).
CopyTemplateLayoutOnly	Boolean used only when TemplateGroupID is not empty. Set to True to copy the existing goup layout only.

Returns	String containing new GroupID (else empty string)
Related Topics	N/A

HasPermission

Exposed In	Version 5.2
Function	Use to determine if the current user has permission to perform a requested function.
Object	Application.BasicFunctions.HasPermission
Syntax	HasPermission (Group as OleVariant, Right as Integer) as Boolean.
Parameters	Group, Right

Group	Right
1 = Contact	1 = Add
2 = Account	2 = Edit
3 = Opportunity	3 = Delete
6 = Ticket	

As an OleVariant, the Group parameter accepts either a Main Table string or the integer values defined in the previous table.

Returns	Returns True if the current user has permission, False if the user does not have permission.
Related Topics	N/A

HelpCurrentView

Function	Specifies the topic called by the help button.
Object	Application.BasicFunctions.HelpCurrentView
Syntax	HelpCurrentView(*aFile : string; aHelpID : integer*)
Parameters	AFile as String. If a file is ("), the current help file is used. HelpID as Integer. If a HelpID is 0, the current help context (pane, main view) is used.
Returns	Boolean
Related Topics	N/A

InsertFileAttachment

Exposed In	Version 6.2
Function	Allows the user to select a file to associate with the current record.
Object	Application.BasicFunctions.InsertFileAttachment
Syntax	InsertFileAttachment()
Parameters	AccountID(String) – The Sage SalesLogix ID of the Account.
Returns	String (the result contains the new AttachmentID, else the result is an empty string)
Related Topics	N/A

InsertFileAttachmentEx

Exposed In	Version 6.2.1
Function	Allows the user to select a file to associate with the current record.
Object	Application.BasicFunctions.InsertFileAttachmentEx
Syntax	InsertFileAttachmentEx()

Parameters	FileName as string AccountID as string
Returns	String (the result contains the new AttachmentID, else the result is an empty string)
Related Topics	N/A

InsertURLAttachment

Exposed In	Version 6.2
Function	Allows the user to associate a URL file (htm, html).
Object	Application.BasicFunctions.InsertUrlAttachment
Syntax	InsertUrlAttachment()
Parameters	AccountID as string
Returns	String (the result contains the new AttachmentID, else the result is an empty string)
Related Topics	N/A

InvokeResult

Function	Returns the result of the last invoke.
Object	Application.BasicFunctions.InvokeResult
Syntax	InvokeResult
Parameters	None
Returns	String
Related Topics	"InvokeSetResult", "InvokeResult"

InvokeSetResult

Function	Sets the resulting value for the current VBScript or legacy Basic invoke. The calling process can read this result by using the "InvokeResult" function.
Object	Application.BasicFunctions.InvokeSetResult
Syntax	InvokeSetResult ()
Parameters	Result as String
Returns	Boolean
Related Topics	"InvokeResult"

InvokeSetStatus

Function	Sets the status code and text for the current VBScript or legacy Basic invoke.
Object	Application.BasicFunctions.InvokeSetStatus
Syntax	InvokeSetStatus *code, status*
Parameters	Code as Long - numerical code number Status as String - text message
Returns	Boolean
Related Topics	"InvokeStatusText", "InvokeStatusCode"

InvokeStatusCode

Function	Returns the status code of the last invoke.
Object	Application.BasicFunctions.InvokeStatusCode
Syntax	InvokeStatusCode
Parameters	None
Returns	Integer
Related Topics	"InvokeSetStatus", "InvokeStatusText", "InvokeStatusCode"

InvokeStatusText

Function	Returns the status text of the last DoInvoke.
Object	Application.BasicFunctions.InvokeStatusText
Syntax	InvokeStatusText
Parameters	None
Returns	String
Related Topics	"InvokeStatusCode"

ISOToDate

Exposed In	Version 6.2
Function	Converts an ISO formatted String (yyyymmdd hh:nn:ss) to a Date/Time format.
Object	Application.BasicFunctions.ISOToDate
Syntax	ISOToDate
Parameters	ISODate as String
Returns	Date Value
Related Topics	N/A

LogAttachSyncRequest

Exposed In	Version 7.0
Function	Allows the user to request an attachment remotely.
Object	Application.BasicFunctions.LogAttachSyncRequest
Syntax	LogAttachSyncRequest (*AttachID*)
Parameters	AttachID as String - KeyFieldID for the attachment
Returns	None
Related Topics	N/A

LogCascadeForTable

Reserved for Sage SalesLogix.

Exposed In	Version 6.2.3

LogCascadeRemove

Reserved for Sage SalesLogix.

Exposed In	Version 6.2.3

LogixClearError

Function	Clears the code and text for an error.
Object	Application.BasicFunctions.LogixClearError
Syntax	LogixClearError
Parameters	None
Returns	String
Related Topics	N/A

LogixErrorCode

Function	Returns the code for a given error.
Object	Application.BasicFunctions.LogixErrorCode
Syntax	LogixErrorCode
Parameters	None
Returns	Integer
Related Topics	“LogixErrors”

LogixErrors

Function	Checks if an error condition has been encountered.
Object	Application.BasicFunctions.LogixErrors
Syntax	LogixErrors
Parameters	None
Returns	Boolean
Related Topics	“LogixErrorCode”

LogixErrorText

Function	Sets the code and text for an error.
Object	Application.BasicFunctions.LogixErrorText
Syntax	LogixErrorText
Parameters	None
Returns	String
Related Topics	“LogixErrors”

LogixSetError

Function	Sets the code and text for an error.
Object	Application.BasicFunctions.LogixSetError
Syntax	LogixSetError ()
Parameters	Code as Integer
	Text as String
Returns	String
Related Topics	N/A

LogSendFileAttachment

Function	Provides synchronization logging of file attachments.
Object	Application.BasicFunctions.LogSendFileAttachment
Syntax	LogSendFileAttachment ()
Parameters	AccountID (String) --Account ID associated with this attachment.
	FileName (String) – File name of a file that exists in the Attachments folder. FileName must be prepended with the site code of the user creating the file attachment.
Returns	Boolean
Related Topics	"LogSetGlobalID"

LogSetGlobalID

Function	This sets the subscription ID (Account ID) used by the Synchronization Server to determine which accounts get records logged from that point on. All records created after LogSetGlobalID is used are only distributed to remote sites that subscribe to the account identified by the ID. This function is handled by the Provider in Sage SalesLogix versions 6.0 and later. However, LogSetGlobalID is still supported for the benefit of those using legacy Basic code.
Object	None
Syntax	LogSetGlobalID ()
Parameters	ID {string}
Returns	Boolean
Related Topics	N/A

LogWhatsNewInsert

Exposed In	Version 6.2
Function	Inserts a Whats New Insert record into the Sage SalesLogix logging file.
Object	Application.BasicFunctions.LogWhatsNewInsert
Syntax	LogWhatsNewInsert
Parameters	Type as String - Main table
	ID as String - Sage SalesLogix ID
Returns	Boolean
Related Topics	N/A

LogWhatsNewInsertAccount

Exposed In	Version 6.2.1
Function	Inserts a What's New Insert record.
Object	Application.BasicFunctions.LogWhatsNewInsertAccount
Syntax	LogWhatsNewInsertAccount()
Parameters	AccountID - string
	SecCodeID - string
	AccountName - string
	City - string
	State - string
	AcctMgr - string

Returns	Boolean
Related Topics	N/A

LogWhatsNewInsertContact

Exposed In	Version 6.2.1
Function	Inserts a Whats New Insert record.
Object	Application.BasicFunctions.LogWhatsNewInsertContact
Syntax	LogWhatsNewInsertContact
Parameters	ContactID - string AccountID - string SecCodeID - string AccountName - string City - string State - string AcctMgr - string LastName - string FirstName - string
Returns	Boolean
Related Topics	N/A

LogWhatsNewSendDoc

Function	Posts a new document notification to What's New in the Sage SalesLogix Remote Client.
Object	Application.BasicFunctions.LogWhatsNewSendDoc
Syntax	LogWhatsNewSendDoc()
Parameters	GlobalID(String) – ID used to distribute the notification through synchronization (See LogSetGlobalID). Key(String) – Key of the attachment record for this document. Description(String) – Description of the document. FileName(String) – File name of the document.
Returns	Boolean
Related Topics	"LogSetGlobalID"

LogWhatsNewUpdate

Function	Posts notification of an updated column to What’s New in the Sage SalesLogix Client.
Object	Application.BasicFunctions.LogWhatsNewUpdate
Syntax	LogWhatsNewUpdate ()
Parameters	aType(String) – Name of the main table.
	aID(String) – Key ID of the updated row.
	aDataPath(String) – Data path of the updated column in the form. TableName: ColumnName.
	aOldVal(String) – Value of the column before update.
	aNewVal(String) – Value of the Column after update
Returns	Boolean
Related Topics	N/A

Example

```
Sub Main
LogWhatsNewUpdate "CONTACT", "CAID0000012", "CONTACT:LASTNAME", "SMITH", "SMITHERS"
End Sub
```

LookUpCalendarUser

Exposed in	Version 7.5
Function	Launch the Find User dialog box based on defaults laoaded using the parameters listed.
Object	Application.BasicFunctions.LookupCalendarUser
	Launches a Lookup based on conditions passed as parameters.
Parameters	OldID as String - the preset USERID property
	OldName as STring - the preset Name property
Returns	An object refernce to the Lookup (ILink).
Related Topics	N/A

LookUpItemWithConditionbyID

Function	Launches a Lookup based on conditions passed as parameters.
Object	Application.BasicFunctions.LookupItemWithConditionByID
Syntax	LookupItemWithConditionByID ()

Parameters

Lookup	(String)	The item to lookup.
RestrictAlways	(Boolean)	
RestrictField	(String)	The field name to be used in the condition.
RestrictValue	(String)	The value of the field specified that is to be restricted.
RestrictOp	(String)	The operand to be used when creating the condition.
InitialText	(String)	

Returns	An OLE variant object with two parameters (DisplayName and ID).

Related Topics	N/A
Example	System:SLX_Lookup_Support

```
'Including Script - System:SLX Error Support
'This function performs a standard Lookup with no conditions and returns the ID
Function LookupByName (strLookupName, strPreFill)
On Error Resume Next
Dim objLookup
Set objLookup = Application.BASICFunctions.LookupItemWithConditionByID (strLookupName,
 False, "", "", "", strPreFill)
  If ErrorCheck ("Error executing Lookup:") > 0 then exit function
  If TypeName (objLookup) = "Link" then  'If type <> "Link" then the user Cancelled
    LookupByName = objLookup.ID
    set objLookup = Nothing
  Else
    LookupByName = -1
  End if
  On Error Goto 0
End function
```

LookUpOwner

Exposed In	Version 6.2
Function	Launches the Find Owner dialog box.
Object	Application.BasicFunctions.LookUpOwner
Syntax	LookUpOwner()
Parameters	SecCode as String - the pre-set seccode of the lookup.
Returns	A String representing the SecCodeID.
Related Topics	N/A

LookUpUser

Function	Launches the Find User dialog box.
Object	Application.BasicFunctions.LookupUser
Syntax	LookupUser
Parameters	None
Returns	An OLE variant object having two parameters (DisplayName and ID) specifying the current user.
Related Topics	N/A

LookUpUserEx

Exposed In	Version 6.2
Function	Launch the Find User dialog box based on defaults loaded using the parameters listed.
Object	Application.BasicFunctions.LookUpUserEx
Syntax	LookUpUserEx()
Parameters	OldID as String - the preset UserID property
	OldName as String - the preset Name property
Returns	An object reference to the Lookup (ILink).
Related Topics	N/A

MergeFromFile

Reserved for Sage SalesLogix.

MergeFromPlugin

Function Behaves the same way that a "Write>E-mail, Letter, Fax...Using Template" action behaves, except that the Most Recently Used (MRU) menu is not changed.

The MRU menu items appear under the Write>...Using Template menus. They only appear there after a merge using the Select a Template dialog, and only if the item does not already appear.

Object Application.BasicFunctions.MergeFromPlugin

Syntax MergeFromPlugin()

Parameters

PluginID (string):	A type 25 plugin ID
MergeMode (integer):	0=E-mail; 1=Fax; 2=Letter
EntityID (string):	A Contact or Lead ID value
OpportunityID (string):	[Optional] An Opportunity associated with the contact
TicketID	[Optional] String used to associate a Ticket with the records created by mail merge.

Returns True for success; False otherwise

Related Topics N/A

MergeFromPluginEx

Function Behaves in the same way as MergeFromPlugin except it adds attachments.

Object Application.BasicFunctions.MergeFromPluginEx

Syntax MergeFromPluginEx()

Parameters

PluginID (string):	A type 25 plugin ID
MergeMode (integer):	0=E-mail; 1=Fax; 2=Letter
EntityID (string):	A Contact or Lead ID value
OpportunityID (string):	[Optional] An Opportunity associated with the contact
AttachIDs	Variant. Array of ATTACHMENT.ATTACHID values.
TicketID	[Optional] String used to associate a Ticket with the records created by mail merge.

Returns True for success; False otherwise

Related Topics N/A

Example

```
Option Explicit
Sub Main
Dim strPluginID, intMergeMode, strContactID, strOpportunityID, colAttachIDs
ReDim colAttachIDs(2)
colAttachIDs(1) = "QQF8AA0009QS"
colAttachIDs(2) = "QQF8AA0009R0"
```

```
strPluginID = "pQF8AA0004AH"
intMergeMode = 0 ' 0=Email; 1=Fax; 2=Letter
strContactID = "CGHEA0002670" ' John Abbott
strOpportunityID = ""
Application.BasicFunctions.MergeFromPluginEx strPluginID, intMergeMode, strContactID,
 strOpportunityID, colAttachIDs
End Sub
```

MergeFromTemplate

Exposed In	Version 6.2
Function	Used to emulate the action that is executed when the user selects the Write>E-mail, Fax, or Letter Using Template>More Templates menu item.
Object	Application.BasicFunctions.MergeFromTemplate
Syntax	MergeFromTemplate()
Parameters	

MergeMode	Long 0 = E-mail 1 = Fax 2 = Letter
EntityID	String
OpportunityID	String
AttachIDs	Variant. Array of ATTACHMENT.ATTACHID values. This parameter is only used if the MergeMode is 0 (E-mail).
AllowSelectOnly	Boolean. If True the user is not be able to create, edit, copy, delete, or preview templates using the Select a Template dialog.
CheckMRU	Boolean. If True the Write>…Using Template most recently used menu is updated if the template has not already been added to the menu.
Canceled	Variant. Returns True if the Select a Template dialog is canceled.
PluginID	Variant. The PluginID of the template the user .
MainTable	Name of the MainTable. (Default = CONTACT)
ShowAllTemplates	If True the Select a Template dialog displays all templates. If False only those templates that are associated with the MainTable are shown. (Default=True)
TicketID	[Optional] String used to associate a Ticket with the records created by mail merge.

Returns	Boolean
Related Topics	N/A

Example

```
Option Explicit
Sub Main
Const errAbort = 25
Const mmEmail = 0
Const mmFax = 1
Const mmLetter = 2
Dim oAttachIDs
Dim bAllowSelectOnly
Dim bCanceled
Dim bCheckMRU
Dim strContactID
Dim strOpportunityID
Dim strPluginID
Dim iMergeMode
iMergeMode = mmLetter
bAllowSelectOnly = False
bCheckMRU = True
If iMergeMode = mmEmail Then
  ReDim oAttachIDs(1)
  oAttachIDs(0) = "QQF8AA0005EW"
  oAttachIDs(1) = "QQF8AA0005F3"
 End If
 strContactID = "CGHEA0002670"
 strOpportunityID = "OQF8AA00001E"
 If Application.BasicFunctions.MergeFromTemplate(iMergeMode, strContactID,
 strOpportunityID, oAttachIDs, bAllowSelectOnly, bCheckMRU, bCanceled, strPluginID)  Then
  ' MsgBox "The following Microsoft Word template plugin was merged to: " & strPluginID
Else
  If bCanceled Then
    ' "The Select a Template dialog was canceled by the user."
  Else
    ' If the merge was not canceled
    If Application.BasicFunctions.GetLastMailMergeErrorType <> errAbort Then
      ' Note: The error will already have been displayed to the user.
      ' MsgBox "There was an error merging. " &
       Application.BasicFunctions.GetLastMailMergeErrorMessage
    Else
      ' "The merge was aborted by the user."
    End If
  End If
End If
End Sub
```

ObjectExists

Function	Checks to see if the object exists.
Object	Application.BasicFunctions.ObjectExists
Syntax	ObjectExists (*objectname*)
Parameters	Object name (string)
Returns	Boolean. Returns True if the object specified exists; False otherwise.
Related Topics	N/A

OpenAttachmentWith

Exposed In	Version 6.2
Function	Allows the user to select a program to use to view an attachment.
Object	Application.BasicFunctions.OpenAttachmentWith
Syntax	OpenAttachmentWith()
Parameters	AttachID as String - KeyField ID for the attachment.
Returns	None
Related Topics	N/A

OverlayDefaultsOnNextView

Function	Puts the system in a state where in the next view shown, the default values for all fields overwrite the current controls. When a view is opened, unbound fields hold the last text string entered as the default value. To override this behavior, use the OverlayDefaultsOnNextView before opening the view. The values for default fields are then set to their default properties.
Object	Application.BasicFunctions.OverlayDefaultsOnNextView
Syntax	OverlayDefaultsOnNextView
Parameters	None
Returns	Boolean
Related Topics	N/A

ParseName

Function	Parses the name passed in to F, L, M, P, or S. Useful for VBScript or legacy Basic imports.
Object	Application.BasicFunctions.ParseName
Syntax	ParseName()
Parameters	Value - The actual name being parsed. This should represent a full name as it appears on the contact NameEdit box.
Returns	A CRLF string
Related Topics	N/A

Example

```
'Including Script - System: SLX Util
Option Explicit
Sub Main
  Dim strNameValues
  Dim strFirstName
  Dim strLastName
  Dim strMiddleName
  Dim strPrefix
  Dim strSuffix
  Dim arrValues
  Dim I

  strNameValues = Application.BasicFunctions.ParseName("Mr. John James Doe III")
  arrValues = Split(Trim(strNameValues), vbCrLf)
  If IsArray(arrValues) Then
    If UBound(arrValues) = 5 Then
      strPrefix = ""
      strFirstName = ""
      strLastName = ""
      strMiddleName = ""
      strSuffix = ""
      For I = LBound(arrValues) To UBound(arrValues)
        Select Case I
        ' First Name
        Case 0
          strFirstName = Trim(arrValues(I))
        ' Last Name
        Case 1
          strLastName = Trim(arrValues(I))
        ' Middle Name
        Case 2
          strMiddleName = Trim(arrValues(I))
        ' Prefix
        Case 3
          strPrefix = Trim(arrValues(I))
        ' Suffix
        Case 4
          strSuffix = Trim(arrValues(I))
        End Select
        Next
        ShowMessage(FormatStr("Prefix: '%s'; FirstName: '%s'; MiddleName: '%s';
          LastName: '%s'; Suffix: '%s'", _
        Array(strPrefix, strFirstName, strMiddleName, strLastName, strSuffix)))
    If
  End If
End Sub
```

PrintAttachment

Exposed In	Version 6.2
Function	Prints the contents of the attachment.
Object	Application.BasicFunctions.PrintAttachment
Syntax	PrintAttachment
Parameters	FileName as String - Full path to the file.
Returns	None
Related Topics	N/A

PrintDetail

-- Caution: Reserved --

Function	Prints a report passed as a parameter for the current record.
Object	Application.BasicFunctions.PrintDetail
Syntax	PrintDetail
Parameters	ReportName as String (the name of the report to be printed.)
Returns	None
Related Topics	N/A

Example

```
'Including Script - System:SLX Error Support
sub Main
'************************************************************************************'
Description: Print Detail View
' Purpose    :
' Called By  : Standard Tool bar
' Calls      :
' Inputs     :
' Outputs    :
' Written    : 09/09/02
' Updates    :
'************************************************************************************D
im sCurrentViewID
Dim sType
Dim oSLX
On Error Resume Next
Set oSLX = Application.BasicFunctions
if ErrorCheck ("Error accessing SalesLogix VBScript functions:") > 0 then exit sub
sCurrentViewID = oSLX.CurrentViewID
sType = Left(sCurrentViewID,1)

'Set the report name to print.
Select case sType
  Case "A"
    oSLX.PrintDetail "Account:Account Detail - Sample"
  Case "C"
    oSLX.PrintDetail "Contact:Contact Detail - Sample"
  Case "O"
    oSLX.PrintDetail "Opportunity:Opportunity Detail - Sample"
  Case Else
    msgBox Localize("Sorry this is not supported on this view")
End Select
if ErrorCheck ("Error Printing Detial:") > 0 then exit sub
End sub
```

ProcessAbort

Function	Stops the current process as soon as possible. This command is only valid in VBScript or in legacy Basic launched from a process.
Object	Application.BasicFunctions.ProcessAbort
Syntax	ProcessAbort
Parameters	None
Returns	Boolean
Related Topics	"ProcessSkipNext"

ProcessSkipNext

Function	Skip the next event in the process. This command is only valid in VBScript or in legacy Basic launched from a process.
Object	Application.BasicFunctions.ProcessSkipNext
Syntax	ProcessSkipNext
Parameters	None
Returns	Boolean
Related Topics	"ProcessAbort"

ProcessWindowMessages

Function	In lengthy operations, allows the application to respond to messages.
Object	Application.BasicFunctions.ProcessWindowMessages
Syntax	ProcessWindowMessages
Parameters	None
Returns	Boolean
Related Topics	N/A

QueMessage

The legacy equivalent of this function is not exposed in the Sage SalesLogix Client.

Function	Populates the relevant parameters and displays the message editor. QueMessage can be by itself or with any of the listed parameters as long as the previous parameter is either filled in or blank. Be aware that some mail clients (or even Mailto:) require an ASCII space for a blank value. The following example shows how "%20" would be used as :
	QueMessage "me@home.com", "%20"
Object	Application.BasicFunctions.QueMessage
Syntax	QueMessage()
Parameters	ToAddress = To Address for the e-mail
	CCAddress = CC Address for the e-mail
	BCCAddress = BCC Address for the e-mail
	Subject = Subject for the e-mail
	Body = Body of the e-mail
	Attach = Attachment path for the e-mail
Returns	Boolean
Related Topics	N/A

The number of characters that can be used successfully in QueMessage is limited. This includes all of the parameters. The sum of your e-mail address, CC, BCC, Subject and Body add up to the limitation, which depends on the operating system.

QueMessageForRecord

Exposed In	Version 6.2.1
Function	Populates the relevant parameters and displays the message editor.
Object	Application.BasicFunctions.QueMessageForRecord
Syntax	QueMessageForRecord()
Parameters	ToAddress = To Address for the e-mail
	CCAddress = CC Address for the e-mail
	BCCAddress = BCC Address for the e-mail
	Subject = Subject for the e-mail
	Body = Body of the e-mail
	Attach = Attachment path for the e-mail
	TableName as string
	KeyFieldValue as string
Returns	Boolean
Related Topics	N/A

QueMessageForRecord

Exposed In	Version 7.0
Function	Displays a list of Contacts associated with a specific Account or Opportunity for the purpose of adding them to the message. If only a single contact exists, then that auto populates the message.
Object	Application.BasicFunctions.QueMessageForRecord
Syntax	QueMessageForRecord()
Parameters	

AttachIDs(optional)	A comma delimited string list of Attachment IDs
Subject(optional)	A string for the subject line of the e-mail.
Body(optional)	A string to be used for the body of the e-mail.

Returns	Boolean
Related Topics	N/A

RefreshActivitiesCache

The number of characters that can be used successfully in QueMessage is limited. This includes all of the parameters. The sum of your e-mail address, CC, BCC, Subject and Body add up to the limitation, which depends on the operating system.

Function	Reloads the activity list used for the Calendar and Activity windows. Use after updating tables with VBScript to synchronize the view with the data.
Object	Application.BasicFunctions.RefreshActivitiesCache
Syntax	RefreshActivitiesCache
Parameters	None
Returns	None
Related Topics	N/A

RefreshHistoryCache

Function	Loads the history cache for the calendar window, but only if it has been previously loaded (meaning that Show Completed is checked on the Calendar). Use after updating history tables with VBScript to synchronize the views with the data.
Object	Application.BasicFunctions.RefreshHistoryCache
Syntax	RefreshHistoryCache
Parameters	None
Returns	None
Related Topics	N/A

RefreshMainView

Function	Reloads the detail portion of the main view, including all forms and tabs. Use after updating tables with VBScript to synchronize the views with the data.
Object	Application.BasicFunctions.RefreshMainView
Syntax	RefreshMainView
Parameters	None
Returns	None
Related Topics	N/A

RegDeletePathValue

Function	Deletes the specified registry value. This command is similar to RegDeleteValue but also allows you to specify the path to the value. A root can be included in the path. The current valid roots are: CLASSES_ROOT, CURRENT_USER, CURRENT_USER, USERS, PERFORMANCE_DATA, CURRENT_CONFIG, and DYN_DATA.
Object	Application.BasicFunctions.RegDeletePathValue
Syntax	RegDeletePathValue (*path, name*)
Parameters	Path as String Name as String
Returns	Boolean
Related Topics	"RegDeleteValue"

RegDeleteValue

Function	Deletes the specified registry value. The registry path used by this command is "CURRENT_USER:Software\SalesLogix\UserValues".
Object	Application.BasicFunctions.RegDeleteValue
Syntax	RegDeleteValue (*name*)
Parameters	Name as String
Returns	Boolean
Related Topics	"RegDeletePathValue"

RegGetPathValue

Function	Similar to RegGetValue, but this function allows you to specify the path to the entry. A root can be included in the path. The current valid roots are: CLASSES_ROOT, CURRENT_USER, CURRENT_USER, USERS, PERFORMANCE_DATA, CURRENT_CONFIG, and DYN_DATA.
Object	Application.BasicFunctions.RegGetPathValue
Syntax	RegGetPathValue (*path, name*)
Parameters	Path as String Name as String
Returns	String
Related Topics	"RegGetValue"

RegGetValue

Function	Returns the current value of the specified registry entry. The registry path used by this function is "CURRENT_USER:Software\SalesLogix\UserValues".
Object	Application.BasicFunctions.RegGetValue
Syntax	RegGetValue (*name*)
Parameters	Name as String
Returns	String
Related Topics	"RegGetPathValue"

RegSetPathValue

Function	Similar to RegSetValue, but this command allows you to specify the path to the entry. A root can be included in the path. The current valid roots are: CLASSES_ROOT, CURRENT_USER, CURRENT_USER, USERS, PERFORMANCE_DATA, CURRENT_CONFIG, and DYN_DATA.
Object	Application.BasicFunctions.RegSetPathValue
Syntax	RegSetPathValue (*path, name, value*)
Parameters	Path as String Name as String Value as String
Returns	None
Related Topics	"RegSetValue"

Example

```
' Set the value Test to Result at the specified path location
Sub Main
Application.BasicFunctions.RegSetPathValue "CURRENT_USER:Software\SalesLogix", "Test",
 "Result"
End Sub
```

RegSetValue

Function	Sets the value of the specified registry entry. The registry path used by this command is "CURRENT_USER:Software\SalesLogix\UserValues".
Object	Application.BasicFunctions.RegSetValue
Syntax	RegSetValue (*name, value*)
Parameters	Name as String

	Value as String
Returns	None
Related Topics	N/A

ReportAddCondition

Function	Adds a condition to the current report using sub-properties to set up the condition that is added. The connector property is always blank for the first condition. For additional conditions, the connector property refers to how the condition relates to the previous condition in the list.
Object	Application.BasicFunctions.ReportAddCondition
Syntax	ReportAddCondition (*datapath, operator, value, type, connector*)
Parameters	Datapath{string} — Name of the field in the database. Operator {string} — Comparison operator. Value {string} — Value to compare. Type {string} — Data type of the field. Connector {string} — Refers to how the previous condition relates to this condition.
Returns	Boolean
Related Topics	"ReportAddConditionEx"

Use the Query Builder to determine the valid values for each parameter. This function always uses case sensitive comparisons.

ReportAddConditionEx

Function	Provides full support for query conditions. Adds a condition to the current report using sub-properties to set up the condition that is added.
Object	Application.BasicFunctions.ReportAddConditionEx
Syntax	ReportAddConditionEx (*datapath, operator, value, type, connector, CaseInSensitive, IsLiteral, Negated*)
Parameters	Datapath {string} - name of the field in the database. Operator {string} - comparison operator. Value {string} - value to compare. Type {string} - data type of the field. Connector {string} - The connector property is always blank for the first condition. For additional conditions, the connector property refers to how the condition relates to the previous condition in the list. CaseInSensitive {Boolean} - indicates whether comparisons should be made using case sensitivity. False indicates that the comparison should be case sensitive. True indicates that the comparison should be case insensitive. IsLiteral {Boolean} - True indicates that the value supplied should be taken as a literal. Useful for date comparisons such as, '03/01/1998'. False indicates that the value supplied is used by the Query Manager to derive a comparison value (for example; within the next xxx days). Negated {Boolean} - True indicates that the result set should be negated (for example; all rows that do not match the condition are returned). False indicates that the result set should not be negated (for example; all rows that match the condition should be returned).

Returns	Boolean
Related Topics	"ReportAddCondition"

Use the Query Builder to determine the valid values for each parameter. This function always uses case sensitive comparisons.

ReportClearConditions

Function	Clears all conditions in a report.
Object	Application.BasicFunctions.ReportClearConditions
Syntax	ReportClearConditions
Parameters	None
Returns	Boolean
Related Topics	N/A

ReportGetConditions

Function	Returns the conditions given for current report.
Object	Application.BasicFunctions.ReportGetConditions
Syntax	ReportGetConditions
Parameters	None
Returns	String
Related Topics	N/A

ReportSetConditions

Function	Sets the conditions for current report.
Object	Application.BasicFunctions.ReportSetConditions
Syntax	ReportSetConditions (*value*)
Parameters	Value as String - a DataPath value. Value uses the following format: |Data Path||operator||Value||Type||IsCaseInsenstive||IsLiteral||Negated|| See "ReportAddConditionEx" on page 54 for more information on each of these values within the pipe-delimited string.
Returns	Boolean
Related Topics	N/A

RunIndexSchedule

Exposed In	Version 7.0
Function	Allows remote users to dynamically run an index schedule.
Object	Application.BasicFunctions.RunIndexSchedule
Syntax	RunIndexSchedule (*aScheduleID*)
Parameters	aScheduleID as String
Returns	None
Related Topics	N/A

RunOpenCloseSchedules

Exposed In	Version 7.0
Function	Allows the user to schedule index rebuilding on application open or application close.
Object	Application.BasicFunctions.RunOpenCloseSchedules
Syntax	RunOpenCloseSchedules(*aOpenSchedules*)
Parameters	aOpenSchedules as String
Returns	None
Related Topics	N/A

SaveAttachmentAs

Exposed In	Version 6.2
Function	Allows the user to save the file using a new file name and location.
Object	Application.BasicFunctions.SaveAttachmentAs
Syntax	SaveAttachmentAs()
Parameters	AttachID as String - KeyFieldID for the attachment
Returns	None
Related Topics	N/A

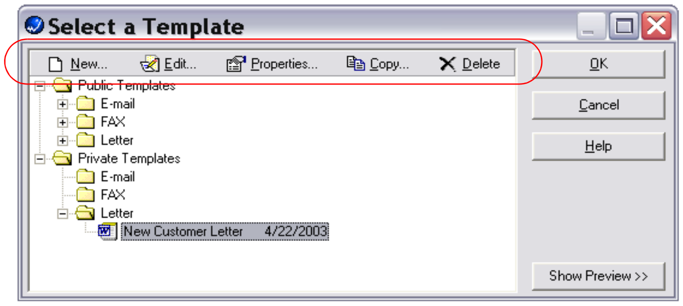
SelectTemplate

Function	Behaves the same way as Write Templates from the Sage SalesLogix Menu.
Object	Application.BasicFunctions.SelectTemplate
Syntax	SelectTemplate
Parameters	None
Returns	Template Name as String. The name of the template.
Related Topics	N/A

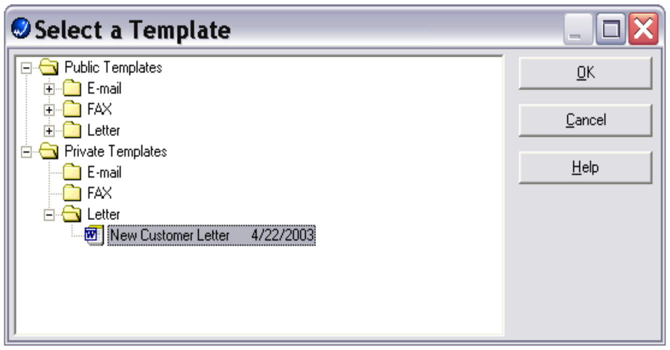
SelectTemplateEx

Exposed In	Version 6.2
Function	Displays the Select a Template dialog. If AllowSelectOnly is False the Show/Hide Preview button and the toolbar are displayed; otherwise, they are hidden.

AllowSelectOnly = False



AllowSelectOnly = True



Object Application.BasicFunctions.SelectTemplateEx

Syntax SelectTemplateEx()

Parameters

AllowSelectOnly	Boolean. If True the user is not be able to create, edit, copy, delete, or preview templates.
Name	Variant. The PLUGIN.NAME of the template.
PluginID	Variant. The PLUGIN.PLUGINID of the template.
Owner	Variant. The PLUGIN.USERID of the template.
Family	Variant. The PLUGIN.FAMILY of the template.
MainTable	Name of the MainTable. (Default = CONTACT)
ShowAllTemplates	If True the Select a Template dialog displays all templates. If False only those templates that are associated with the MainTable are shown. (Default=True)

Returns Boolean

Related Topics N/A

Example

```
Option Explicit
Sub Main
Const errAbort = 25
Const mmEmail = 0
Const mmFax = 1
Const mmLetter = 2
Dim oAttachIDs
Dim bAllowSelectOnly
Dim strContactID
Dim strFamily
Dim strName
Dim strOpportunityID
Dim strOwner
Dim strPluginID
Dim iMergeMode
iMergeMode = mmEmail
bAllowSelectOnly = False
If Application.BasicFunctions.SelectTemplateEx(bAllowSelectOnly, strName, strPluginID,
 strOwner, strFamily) Then
    ' The user  a template
    ' Check the Write>...Using Template Most Recently Used (MRU) menu items
    If Not Application.BasicFunctions.CheckWriteMRUMenu(strPluginID, iMergeMode) Then
      Exit Sub
    End If
    strContactID = "CGHEA0002670"
    strOpportunityID = "OQF8AA00001E"
    If iMergeMode = mmEmail Then
      ' Add attachments (mmEmail only)
      ReDim oAttachIDs(1)
      oAttachIDs(0) = "QQF8AA0005EW"
      oAttachIDs(1) = "QQF8AA0005F3"
    End If
    ' Execute the mail merge
    If Application.BasicFunctions.MergeFromPluginEx(strPluginID, iMergeMode, strContactID,
     strOpportunityID, oAttachIDs) Then
      ' The merge was successful
    Else
      ' If the merge was not canceled
      If Application.BasicFunctions.GetLastMailMergeErrorType <> errAbort Then
        ' Note: The error will already have been displayed to the user.
        ' MsgBox "There was an error merging. " &
         Application.BasicFunctions.GetLastMailMergeErrorMessage
      Else
        ' "The merge was canceled by the user."
      End If
    End If
  Else
    ' "The Select a Template dialog was canceled by the user."
  End If
End Sub
```

SetCurrentAccountID

Function	Switches the current client view to "Accounts" and makes the account identified in the AccountID parameter visible within the view.
Object	Application.BasicFunctions.SetCurrentAccountID
Syntax	SetCurrentAccountID (*AccountID*)
Parameters	AccountID as String - The 12 character account ID.
Returns	None
Related Topics	N/A

The AccountID string is the record key for the account table.

SetCurrentClientGroup

Function	Sets the named Group as the current group.
Object	Application.BasicFunctions.SetCurrentClientGroup
Syntax	SetCurrentClientGroup *GroupName*
Parameters	GroupName(String)
Returns	Boolean
Related Topics	"CreateTempAdHocGroup"

SetCurrentContactID

Function	Switches the current client view to "Contacts" and makes the contact identified in the ContactID parameter visible within the view.
Object	Application.BasicFunctions.SetCurrentContactID
Syntax	SetCurrentContactID *ID*
Parameters	ID as String - The 12 character contact ID.
Returns	Boolean
Related Topics	N/A

The ContactID string is the record key for the Contact table.

SetCurrentOpportunityID

Function	Switches the current client view to "Opportunities" and makes the opportunity identified in the OpportunityID parameter visible within the view.
Object	Application.BasicFunctions.SetCurrentOpportunityID
Syntax	SetCurrentOpportunityID *ID*
Parameters	ID as String - The 12 character opportunity ID.
Returns	Boolean
Related Topics	N/A

The OpportunityID string is the record key for the Opportunity table.

SetCurrentViewCancelCaption

Function	Replaces the default caption of the Cancel button on a view.
Object	Application.BasicFunctions.SetCurrentViewCancelCaption
Syntax	SetCurrentViewCancelCaption *(Caption)*
Parameters	Caption(String) – Caption for the Cancel button.
Returns	Boolean
Related Topics	"SetCurrentViewOKCaption", "SetCurrentViewHelpCaption"

SetCurrentViewCaption

Function	Replaces the default caption for a particular view. This function is useful for data views that obtain caption properties from the Edit actions taken from a DataGrid control.
Object	Application.BasicFunctions.SetCurrentViewCaption
Syntax	SetCurrentViewCaption (*Caption)*
Parameters	Caption(String) –Text string representing the caption to be used for the view.
Returns	Boolean
Related Topics	N/A

SetCurrentViewHelpCaption

Function	Replaces the default caption of the Help button on a view.
Object	Application.BasicFunctions.SetCurrentViewHelpCaption
Syntax	SetCurrentViewHelpCaption *(Caption)*
Parameters	Caption(String) – Caption for the Help button.
Returns	Boolean
Related Topics	"SetCurrentViewOKCaption", "SetCurrentViewCancelCaption"

SetCurrentViewOKCaption

Function	Replaces the default caption of the OK (or Close) button on a view.
Object	Application.BasicFunctions.SetCurrentViewOKCaption
Syntax	SetCurrentViewOKCaption *(Caption)*
Parameters	Caption(String) – Caption for the OK button.
Returns	Boolean
Related Topics	"SetCurrentViewCancelCaption", "SetCurrentViewHelpCaption"

SetDataPathValue

Function	Sets the data path to the value passed in by aNewValue.
Object	Application.BasicFunctions.SetDataPathValue
Syntax	SetDataPathValue *datapath, anewvalue*
Parameters	DataPath as String - Table:Field, for example: Contact:LastName
	aNewValue as String - for example, Jones
Returns	String
Related Topics	N/A

SetDelimitedTerm

Function	Returns a delimited string that contains the source string properly delimited by the specified character delimiter. This function, in conjunction with GetDelimitedTerm is useful for manipulating multiple values in a single string variable.
Object	Application.BasicFunctions.SetDelimitedTerm
Syntax	SetDelimitedTerm(*Value, Index, Source, Delimiter*)

Parameters	Value (String) – delimited string. Index (Integer) – position of item within the delimited string. Source (String) – item to be inserted into delimited string. Delimiter (String) – character used to delimit string. The delimiter can be any single character and, if left blank, is "|".
Returns	String – A delimited string
Related Topics	"GetDelimitedTerm", "GetDelimitedTermCount"

SetPassword

Function	Allows for the modification of passwords during runtime of the Sage SalesLogix Client. A user is only allowed to make changes to personal passwords. If a user attempts to modify a password for another, an error code is returned and the change is not be made. The user login ADMIN can make a modification to any user login.
Object	Application.BasicFunctions.SetPassword
Syntax	Setpassword(UserLogin{*as string*}, NewPassword{*as string*}
Parameters	UserLogin - String Value detailing the user code (case sensitive) that you want to make the password change to. For example, "ADMIN", "lee" NewPassword - String Value detailing the new password.
Returns	String - 4 possible returns: If everything runs correctly, a blank string is returned. If the password write to the database fails, the following error message is returned: 'Error! Unable to set password.' If the password fails password verification, the following error message is returned: 'Error! Password fails verification options'. If a user is attempting to set a password for another user, the following error message is returned: 'Error! You are not allowed to change passwords of other users'.
Related Topics	N/A

SetTabVisibleProperty

Function	Allows a script to check or change visible status of a Tab. For example, you might use this function to check to see if tabs are visible that should be, and vice versa.
Object	Application.BasicFunctions.SetTabVisibleProperty
Syntax	SetTabVisibleProperty(*Name, Value)*
Parameters	Name - String value detailing which tab you would like to check or change visibility status for. Value as Boolean.
Returns	Boolean
Related Topics	N/A

Example

```
Application.BasicFunctions.SetTabVisibleProperty "Tab Caption", false
Application.BasicFunctions.SetTabVisibleProperty "Personal:Plugin Name", false
```

ShowActivity

Exposed In	Version 6.2
Function	Launches the Schedule Activity dialog for the Activity ID passed as a parameter.
Object	Application.BasicFunctions.ShowActivity
Syntax	ShowActivity
Parameters	Activity ID as String
Returns	None
Related Topics	N/A

ShowActivityNotePad

Exposed In	Version 6.2
Function	Shows the Activity Note Pad for the supplied Activity ID.
Object	Application.BasicFunctions.ShowActivityNotePad
Syntax	ShowActivityNotePad (*ActivityID)*
Parameters	ActivityID (String) – a valid Activity Identifier
Returns	Boolean
Related Topics	"EditActivity", "CompleteActivity"

ShowActivityNotePadEx

Exposed In	Version 6.2
Function	Shows the Activity Notepad dialog box pre-populated with values passed as parameters.
Object	Application.BasicFunctions.ShowActivityNotePadEx
Syntax	ShowActivityNotePadEx()
Parameters	aActivityID as String - a valid Activity identifier. aRegarding as String - text to appear in the Regarding text dialog box. aNotes as String - text to appear in the Notes memo dialog box. ContactID as String AccountID as String OpportunityID as String TicketID as String
Returns	None
Related Topics	N/A

ShowAddForm

Exposed In	Version 7.0
Function	Adds a new record to the specified table. The record is first displayed in the specified form.
Object	Application.BasicFunctions.ShowAddForm
Syntax	ShowAddForm()
Parameters	TableName as String
	ViewName as String
	RecordID as String
	BindDataPath as Variant
	BindValue as Variant
Returns	New record’s key (or an empty string if failed or canceled).
Related Topics	N/A

ShowCalenderReports

Exposed In	Version 6.2
Function	Launches the Calendar Report options dialog from which the user can print and preview.
Object	Application.BasicFunctions.ShowCalenderReports
Syntax	ShowCalenderReports
Parameters	None
Returns	Boolean
Related Topics	N/A

ShowDefaultGroup

Exposed In	Version 7.0
Function	Returns the Plugin ID for the Default Group.
Object	Application.BasicFunctions.ShowDefaultGroup
Syntax	ShowDefaultGroup()
Parameters	MainTable (String) - table name for main view of the default group.
Returns	String containing Default Group plugin ID (else empty string)
Related Topics	N/A

ShowDetails

Function	Displays the Detail view
Object	Application.BasicFunctions.ShowDetails
Syntax	ShowDetails (*table*, *ID*)
Parameters	TableName as String - the main table for the form.
	ID as String - the Sage SalesLogix ID for the main table.
Returns	None
Related Topics	N/A

ShowHistory

Exposed In	Version 5.2
Function	Displays the Show History dialog for the HISTORY.HISTORYID passed.
Object	Application.BasicFunctions.ShowHistory
Syntax	ShowHistory()
Parameters	HistoryID
Returns	String
Related Topics	N/A

ShowMainViewFromLookupWithConditionByID

Function	Shows the MainView from a Lookup based on conditions passed as parameters.
Object	Application.BasicFunctions.ShowMainViewFromLookupWithConditionByID
Syntax	ShowMainViewFromLookupWithConditionByID()
Parameters	

Lookup	(String)	The item to lookup.
RestrictAlways	(Boolean)	
RestrictField	(String)	The field name to be used in the condition.
RestrictValue	(String)	The value of the field specified that is to be restricted.
RestrictOp	(String)	The operand to be used when creating the condition.
InitialText	(String)	

Returns	String
Related Topics	N/A

ShowReports

Exposed In	Version 6.2
Function	Shows the Report Manager with Family expanded.
Object	Application.BasicFunctions.ShowReports
Syntax	ShowReports
Parameters	ReportType as String - the family for the report.
Returns	None
Related Topics	N/A

ShowSearchOptions

Exposed In	Version 7.0
Function	Shows the SpeedSearch Options dialog.
Object	Application.BasicFunctions.ShowSearchOptions
Syntax	ShowSearchOptions
Parameters	None
Returns	None
Related Topics	N/A

ShowViewForRecord

Function	Shows a named plugin view for a specific record number. An example of this would be to show the Ticket Activity View for a specific Activity number. The record number must be a valid primary record number for the view. This function would not be useful for Grid or List views as you would use DoInvoke instead.
Object	Application.BasicFunctions.ShowViewForRecord
Syntax	ShowViewForRecord
Parameters	TableName as String - the main table for the form. For example, Ticket Activity.
	ViewName as String - A full plugin view name, for example "SYSTEM:Support Ticket Acitvity Popup"
	RecordID as String - a Key Field ID for the main table of the view being displayed. For example, " TicketActivity ID would be "a43YI25978324."
Returns	An integer specifying if the user clicked OK or Cancel.
Related Topics	N/A
Example	See the Contact:Support Contact Ticket DblClick script.

ShowViewForRecordEx

Function	Shows a named plugin view for a specific record number.
Object	Application.BasicFunctions.ShowViewForRecordEx
Syntax	ShowViewForRecordEx
Parameters	TableName as String
	ViewName as String
	RecordID as String
	BindDataPath as Variant
	BindValue as Variant
Returns	Long
Related Topics	N/A

StartContactProcess

Function	Triggers the Start Contact Process dialog. You can supply default values for Contact, Group or Opportunity, and Process Name.	
Object	Application.BasicFunctions.StartContactProcess	
Syntax	StartContactProcess()	
Parameters		
	EntityID as String:	Contact
		Opportunity
		Group
	EntityIDType as String:	If EntityIDType = Contact Then ContactID
		If EntityIDType = Opportunity Then OpportunityID
		If EntityIDType = Group Then GroupID
	ProcessName as String:	"Family:PluginName"
Returns	Boolean	
Related Topics	N/A	

StringToColor

Function	Converts a color name or hexadecimal value into a RGB value.	
	Basic colors	Black, Maroon, Green, Olive, Navy, Purple, Teal, Gray, Silver, Red, Lime, Yellow, Blue, Fuchsia, Aqua, White
	System colors (Defined by Start/Settings/Control Panel/Display/ Appearance).	ScrollBar, Background, ActiveCaption, InactiveCaption, Menu, Window, WindowFrame, MenuText, WindowText, CaptionText, ActiveBorder, InactiveBorder, AppWorkSpace, Highlight, HighlightText, BtnFace, BtnShadow, GrayText,BtnText, InactiveCaptionText, BtnHighlight, 3DDkShadow, 3DLight, InfoText, InfoBk
Object	Application.BasicFunctions.StringToColor	
Syntax	StringToColor	
Parameters	Value as String	
Returns	Color code integer	
Related Topics	N/A	

For localization: the string can be in ***English*** only and returns the color code integer. Any other language will not work.

Subscribe

Function	Used to subscribe a user to an account by writing to the log.
Object	Application.BasicFunctions.Subscribe
Syntax	Subscribe
Parameters	AccountID as String - the Sage SalesLogix ID for the account.
	UserID as String - the Sage SalesLogix ID for the User.
Returns	Boolean
Related Topics	N/A

SystemInfoExists

Function	Used to determine if a system global identifier exists in the current instance of the Client. This function is useful for determining if the system variable you want to retrieve is supported in the release of the executing Client.
Object	Application.BasicFunctions.SystemInfoExists
Syntax	SystemInfoExists(*ID*)
Parameters	ID as String – Identifier of the System global identifier
Returns	String
Related Topics	"SystemInfoFor"

SystemInfoFor

Function	Used to obtain a system global identifier from the current instance of the Client.
Object	Application.BasicFunctions.SystemInfoFor
Syntax	SystemInfoFor(*Ident*)
Parameters	Ident(String) – Name of the System global identifier. See the following table for a list of available identifiers.

Identifiers	**Returns**
BuildNumber	Build number of the Client.
CurrentAccountID	Returns the ID of the current Account in the Client.
CurrentAccountName	Returns the account name of the current Account in the Client.
CurrentContactID	Returns the ID of the current Contact in the Client.
CurrentOpportunityID	Returns the ID of the current Opportunity in the Client.
CurrentViewID	Returns the ID for the active view. If a managed view, then returns the ID of the row for the first grid found. If the view is currently adding a record (rather than editing) the returned ID is blank (" ").
DatabaseAlias	Returns the name of the database alias from the current connection string.
DatabaseNames	Returns the database file system name from the SystemInfo Object.
DatabasePath	Returns the database name.
DatabaseServerName	Returns the computer name where the database server is running.
DatabaseType	Returns the database type from the Sage SalesLogix Provider.
DatabaseUsername	Returns the database username in the Connection Manager on the Sage SalesLogix Server for the user logged in as "SYSDBA".
DefaultAreaCode	Returns the default area code for the user (e.g., (602)).
DefaultSecCodeID	Returns the default security code ID.
Lexicon	Returns a CRLF delimited list of identifiers supported by the SystemInfoFor command.
MajorVersion	Client major version (as in 3).
MinorVersion	Client minor version (as in 0).
PlatformID	Returns the operating system platform ID.
PlatformIsNT	Returns True if the platform is NT.
SLXServerName	Returns the computer name where the Sage SalesLogix Server is running.
SLXServerPort	Returns the port number currently being used.
SLXServerList	Returns a comma delimited list of Sage SalesLogix Servers running under the current port.
SLXUsername	Returns the Sage SalesLogix Login name.

Identifiers	Returns
SLXServerAliasList	Returns a comma delimited list of aliases that were created in the Connection Manager on the current Sage SalesLogix Server.
SLXClientAliasList	Returns a comma delimited list of aliases that were created in the Data Link Manager on the Client computer.
SLXExternalADOAliasList	Returns a comma delimited list of aliases that were stored in the AliasAdo.ini file. These are aliases that have a connection string that connects to an external data source.
UserID	Returns the user ID of the person logged on to the Client.
Username	Returns the username of the person logged on to the Client.

Returns String

Related Topics "SystemInfoExists"

SystemInfoSet

Function Use to set a system global identifier from the current instance of the Client. This function uses the same verbs as SystemInfoFor. It can only be used if SystemInfoSettable returns true.

Object Application.BasicFunctions.SystemInfoSet

Syntax SystemInfoSet

Parameters ID equals the identifier of the SystemInfo item to set. See SystemInfoFor for a list of identifiers.

Value equals the value to be assigned to that ID.

Returns True if the value is assigned successfully, otherwise returns False.

Related Topics N/A

SystemInfoSettable

Function Use this function in conjuction with SystemInfoSet to return true if System Info verb can be set.

Object Application.BasicFunctions.SystemInfoSettable

Syntax SystemInfoSettable

Parameters ID as String - equals the identifier of the SystemInfo item.

Returns True if the systemInfo object (or global variable) can be written to, otherwise returns False.

Related Topics N/A

TzCalculateTimeZoneDateTime

Exposed In	Version 6.2
Function	This method displays the “Time Zone Calculator” dialog, allowing the user to define a date and time in relationship to another time zone.
Object	Application.BasicFunctions.TzCalculateTimeZoneDateTime
Syntax	TZCalculateTimeZoneDateTime
Parameters	

TimeZoneKey As String	The time zone identifier. For example: “Mexico Standard Time”. The TimeZoneKey parameter is used to retrieve the “Comparison Time Zone” display name. Default is the current time zone.
DateTime As Date	The DateTime parameter is used to display the date and time for the “Current Time Zone.” Default is the current date and time.
DateTimeMessage As String	The DateTimeMessage parameter is used to display the first line of the date and time message. Default is “Activity will be scheduled for:”.
DateTime As Variant	

Returns	If the user clicks OK, TzCalculateTimeZoneDateTime returns True and the DateTime parameter contains the date and time the user has .
Related Topics	N/A

Example

```
Dim strDateTimeMessage
Dim strTimeZoneKey
Dim dtDateTime
Dim dtScheduledDateTime

strDateTimeMessage = "The  date/time is:"
strTimeZoneKey = "AUS Eastern Standard Time"
dtScheduledDateTime = "12/25/2003 4:00:00 PM"
If Application.BasicFunctions.TzCalculateTimeZoneDateTime(strTimeZoneKey,
dtScheduledDateTime, strDateTimeMessage, dtDateTime) Then
  MsgBox "Scheduled for: " & dtDateTime
Else
  ' The dialog was canceled.
End If
```

TzDateFallsWithinDaylightTime

Exposed In	Version 6.2
Function	This method is used to return whether or not the LocalDate falls within Daylight Saving Time for the time zone represented by the TimeZoneKey. The LocalDate is adjusted according to the rules of the DaylightAdjustment parameter.
Object	Application.BasicFunctions.TzDateFallsWithinDaylightTime
Syntax	TzDateFallsWithinDaylightTime
Parameters	LocalDate As Date - The date that is local to the time zone represented by the TimeZoneKey value.

TimeZoneKey As String - Used to retrieve the "Comparison Time Zone" display name. Default is the current time zone.

DaylightAdjustment As DaylightAdjustmentKind.

The DaylightAdjustmentKind parameter defines how a value is treated in relation to Daylight Savings Time. If a time zone is not currently observing Daylight Savings Time then changing this parameter has no effect.The following DaylightAdjustmentKind enumeration values have been defined:

daAutoAdjustment	Value = 0	If the TimeZoneKey represents the local system time zone TzDateFallsWithinDaylightTime behaves the same way it does with the daForceAdjustment option if the "Automatically adjust clock for daylight savings changes" option on the "Date and Time Properties" control panel has been checked. It behaves the same as it does with the daForceNoAdjustment option if it has been unchecked. If the TimeZoneKey does NOT represent the local system time zone then TzDateFallsWithinDaylightTime behaves the same way it does with daForceAdjustment.
daForceNoAdjustment	Value = 1	The result is always False. This option causes TzDateFallsWithinDaylightTime to act as if the "Automatically adjust clock for daylight savings changes" option on the "Date and Time Properties" control panel has been unchecked; effectively disabling Daylight Saving Time altogether.
daForceAdjustment	Value = 2	The result is True if the time zone observes Daylight Saving Time and the date falls within Daylight Saving Time. This option causes TzDateFallsWithinDaylightTime to act as if the "Automatically adjust clock for daylight savings changes" option on the "Date and Time Properties" control panel has been checked.

Returns Boolean

Related Topics N/A

Example See "TzTimeZoneObservesDaylightTime" on page 81.

TzGetAddressTimeZoneKey

Exposed In	Version 6.2
Function	This method is used to return the TimeZoneKey stored in the ADDRESS.TIMEZONE field for a given AddressID.
Object	Application.BasicFunctions**.**TzGetAddressTimeZoneKey()
Syntax	TzGetAddressTimeZoneKey()
Parameters	AddressID As String - a valid ADDRESSID value from the ADDRESS table.
Returns	String
Related Topics	N/A

Example

```
Dim strDisplayName
Dim strTimeZoneKey
strTimeZoneKey = Application.BasicFunctions.TzGetAddressTimeZoneKey("aA2EK0011631")
If strTimeZoneKey <> "" Then
  strDisplayName = Application.BasicFunctions.TzTranslateTimeZoneKey(strTimeZoneKey)
  MsgBox "The current time for " & strDisplayName & " is: " &
 Application.BasicFunctions.TzGetCurrentLocalDateTimeForTimeZone(strTimeZoneKey,
 daForceAdjustment)
Else
  MsgBox "The address does not have a time zone defined."
End If
```

TzGetConnectionDaylightAdjustment

Exposed In	Version 6.2
Function	This method is used to retrieve the DaylightAdjustmentKind value associated with the OLE DB provider’s database connection. The daAutoAdjustment value is the only value currently supported.
Object	Application.BasicFunctions.TzGetConnectionDaylightAdjustment
Syntax	TzGetConnectionDaylightAdjustment
Parameters	None
Returns	A DaylightAdjustmentKind value
Related Topics	N/A

TzGetConnectionTimeZoneKey

Exposed In	Version 6.2
Function	This method is used to retrieve the TimeZoneKey associated with the Sage SalesLogix OLE DB provider’s TIMEZONE extended connection property.
Object	Application.BasicFunctions.TzGetConnectionTimeZoneKey
Syntax	TzGetConnectionTimeZoneKey
Parameters	None
Returns	A String value. If the TIMEZONE extended property is not defined or is defined incorrectly, the TimeZoneKey associated with the local system is returned (see also TzGetLocalSystemTimeZoneKey). If the TIMEZONE property is “NONE” then “NONE” is the value returned.
Related Topics	N/A

TzGetCurrentLocalDateTimeForTimeZone

Exposed In	Version 6.2
Function	This method is used to return the current local date for the time zone represented by the TimeZoneKey. The result is adjusted according to the rules of the DaylightAdjustment parameter.
Object	Application.BasicFunctions.TzGetCurrentLocalDateTimeForTimeZone()
Syntax	TzGetCurrentLocalDateTimeForTimeZone()
Parameters	TimeZoneKey As String - The time zone identifier. For example: "Mexico Standard Time". DaylightAdjustment As DaylightAdjustmentKind (for more information about using DaylightAdjustmentKind, see "TzDateFallsWithinDaylightTime" on page 69.)
Returns	A Date value
Related Topics	N/A
Example	See "TzGetAddressTimeZoneKey" on page 71.

TzGetCurrentLocalDateTimeForTimeZoneAsString

Exposed In	Version 6.2
Function	This method is used to return the current local date for the time zone represented by the TimeZoneKey. The date is adjusted according to the rules of the DaylightAdjustment parameter and formatted according to the rules of the FormatString (see Architect and Query Builder Help).
Object	Application.BasicFunctions.TzGetCurrentLocalDateTimeForTimeZoneAsString()
Syntax	TzGetCurrentLocalDateTimeForTimeZoneAsString()
Parameters	TimeZoneKey As String --The time zone identifier. For example: "Mexico Standard Time". DaylightAdjustment As DaylightAdjustmentKind (for more information about using DaylightAdjustmentKind, see "TzDateFallsWithinDaylightTime" on page 69.) FormatString As String - A value used to format the date returned. See the Architect Help topic "Date/Time Field Format Strings" for more information.
Returns	String
Related Topics	N/A

TzGetDaylightName

Exposed In	Version 6.2
Function	Use this method to retrieve the DaylightName for the time zone represented by the TimeZoneKey. For example, "AUS Eastern Daylight Time", "Central Daylight Time".
Object	Application.BasicFunctions.TzGetDaylightName()
Syntax	TzGetDaylightName()
Parameters	TimeZoneKey As String - The time zone identifier. For example: "Mexico Standard Time". DaylightName As Variant -- The daylight name. For example: "Mexico Daylight Time". The DayllightName is a string associated with daylight time on the operating system.
Returns	TzGetDaylightName returns True if the DaylightName was retrieved and False if the TimeZoneKey is invalid.

Related Topics N/A

Example

```
Sub Main
Dim strDisplayName
Dim strTimeZoneDescription
Dim strTimeZoneKey
Dim dtSomeDateTime

'strTimeZoneKey = "AUS Eastern Standard Time"
strTimeZoneKey = "US Mountain Standard Time"
strDisplayName = Application.BasicFunctions.TzTranslateTimeZoneKey(strTimeZoneKey)
dtSomeDateTime = "6/7/2004 8:00:00 AM"
'dtSomeDateTime = "12/12/2004 7:30:00 AM"
strTimeZoneDescription = GetTimeZoneDescription(strTimeZoneKey, dtSomeDateTime,
 strDisplayName)
MsgBox strTimeZoneDescription
End Sub

Function GetTimeZoneDescription(ATimeZoneKey, ADateTime, ADefault)
Dim strDaylightName
Dim strStandardName
GetTimeZoneDescription = ADefault
If Application.BasicFunctions.TzIsValidTimeZoneKey(ATimeZoneKey) Then
  If Application.BasicFunctions.TzTimeZoneObservesDaylightTime(ATimeZoneKey) Then
    If Application.BasicFunctions.TzDateFallsWithinDaylightTime(ADateTime, ATimeZoneKey,
     0) Then
      If Application.BasicFunctions.TzGetDaylightName(ATimeZoneKey, strDaylightName) Then
        GetTimeZoneDescription = strDaylightName
      End If
      Else
        If Application.BasicFunctions.TzGetStandardName(ATimeZoneKey, strStandardName) Then
         GetTimeZoneDescription = strStandardName
      End If
    End If
  Else
    If Application.BasicFunctions.TzGetStandardName(ATimeZoneKey, strStandardName) Then
      GetTimeZoneDescription = strStandardName
    End If
  End If
End If
End Function
```

TzGetLocalSystemTimeZoneKey

Exposed In	Version 6.2
Function	This method is used to return the TimeZoneKey for the local system.
Object	Application.BasicFunctions.TzGetLocalSystemTimeZoneKey
Syntax	TzGetLocalSystemTimeZoneKey
Parameters	None
Returns	A String value. For example, if the current time zone is “(GMT-07:00) Arizona” then the returned value would be “US Mountain Standard Time”.
Related Topics	N/A
Example	See “TzTimeZoneObservesDaylightTime” on page 81.

TzGetStandardName

Exposed In	Version 6.2
Function	This method is used to retrieve the StandardName for the time zone represented by the TimeZoneKey. For example, "AUS Eastern Standard Time", "Central Standard Time", and so on.
Object	Application.BasicFunctions.TzGetStandardName()
Syntax	TzGetStandardName()
Parameters	TimeZoneKey As String -- The time zone identifier. For example: "Mexico Standard Time". StandardName As Variant -- The standard name. For example: "Mexico Standard Time". The StandardName is a string associated with standard time on the operating system.
Returns	TzGetStandardName returns True if the StandardName was retrieved and False if the TimeZoneKey is invalid.
Related Topics	N/A
Example	See "TzGetDaylightName" on page 72.

TzGetTimeZoneInformation

Exposed In	Version 6.2
Function	This method is used to retrieve information from the registry for the time zone represented by the TimeZoneKey.
Object	Application.BasicFunctions.TzGetTimeZoneInformation()
Syntax	TzGetTimeZoneInformation()
Parameters	

TimeZoneKey	String: The time zone identifier. For example: "Mexico Standard Time".
Year	Long: The year the StandardStartDate and DaylightStartDate should be calculated for.
DisplayName	Variant: The time zone display name. For example: "(GMT-06:00) Guadalajara, Mexico City, Monterrey".
Bias	Variant: The time zone bias. For example: "360". The bias is the difference, in minutes, between GMT and local time.
StandardName	Variant: The standard name. For example: "Mexico Standard Time". The StandardName is a string associated with standard time on the operating system.
StandardBias	Variant: The standard bias. For example: "0". The StandardBias is the value is to be used during local time translations that occur during standard time. This value is added to the value of the Bias parameter to form the bias used during standard time. In most time zones, the value of this parameter is zero. This value should be ignored if the value for the StandardStartDate has not been defined.

StandardStartDate	Variant: The start date for standard time. For example: “9/28/2003 2:00:00 AM”. The StandardStartDate is the date and local time when the transition from daylight saving time to standard time occurs on the operating system.
DaylightName	Variant: The daylight name. For example: “Mexico Daylight Time”. The DayllightName is a string associated with daylight time on the operating system.
DaylightBias	Variant: The daylight bias. For example: “-60”. The DaylightBias is the value is to be used during local time translations that occur during daylight saving time. This value is added to the value of the Bias parameter to form the bias used during daylight saving time. In most time zones, the value of this parameter is -60. This value should be ignored if the value for the DaylightStartDate has not been defined.
DaylightStartDate	Variant: The start date for daylight time. For example: “5/4/2003 2:00:00 AM”. The DaylightStartDate is the date and local time when the transition from standard time to daylight saving time occurs on the operating system.

Returns TzGetTimeZoneInformation returns True if the information was retrieved and False if the TimeZoneKey is invalid.

Related Topics N/A

Example

```
Dim strTimeZoneKey
Dim strDisplayName
Dim strStandardName
Dim strDaylightName
Dim dtStandardStartDate
Dim dtDaylightStartDate
Dim iYear
Dim iBias
Dim iStandardBias
Dim iDaylightBias

strTimeZoneKey = "AUS Eastern Standard Time"
'strTimeZoneKey = "US Mountain Standard Time"
iYear = 2004
If Application.BasicFunctions.TzIsValidTimeZoneKey(strTimeZoneKey) Then
  If Application.BasicFunctions.TzGetTimeZoneInformation(strTimeZoneKey, iYear,
 strDisplayName, iBias, strStandardName, iStandardBias, dtStandardStartDate,
 strDaylightName, iDaylightBias, dtDaylightStartDate) Then
  If Application.BasicFunctions.TzTimeZoneObservesDaylightTime(strTimeZoneKey) Then
    MsgBox "Display Name: " & strDisplayName & vbCrLf & _
      "Bias: " & iBias & vbCrLf & _
      "Standard Name: " & strStandardName & vbCrLf & _
      "Standard Bias: " & iStandardBias & vbCrLf & _
      "Standard Start Date: " & dtStandardStartDate & vbCrLf & _
      "Daylight Name: " & strDaylightName & vbCrLf & _
      "Daylight Bias: " & iDaylightBias & vbCrLf & _
      "Daylight Start Date: " & dtDaylightStartDate
  Else
```

```
        MsgBox "Display Name: " & strDisplayName & vbCrLf & _
          "Bias: " & iBias & vbCrLf & _
          "Standard Name: " & strStandardName & vbCrLf & _
          "Standard Bias: N/A" & vbCrLf & _
          "Standard Start Date: N/A" & vbCrLf & _
          "Daylight Name: N/A" & vbCrLf & _
          "Daylight Bias: N/A" & vbCrLf & _
          "Daylight Start Date: N/A"
      End If
    End If
    End If
```

TzGetUserTimeZoneKey

Exposed In	Version 6.2
Function	This method is used to return the TimeZoneKey stored in the USERINFO.TIMEZONE field for a given UserID. The TimeZoneKey value that is returned may or may not correspond to the current time zone for that user. Initially, the value that is in this field is defined when the Sage SalesLogix database is converted to GMT using the "Sage SalesLogix Date/Time to GMT Converter" application. The USERINFO.TIMEZONE field value changes based on the local system time zone that was in effect the last time a user logged in.
Syntax	Application.BasicFunctions.TzGetUserTimeZoneKey()
Parameters	UserID As String -- The Sage SalesLogix ID for the user.
Returns	String
Related Topics	N/A

TzGMTToLocal

Exposed In	Version 6.2
Function	This method is used to convert the GMTDate into a local date for the current time zone. The result is adjusted according to the rules of the DaylightAdjustment parameter.
Syntax	Application.BasicFunctions.TzGMTToLocal()
Parameters	GMTDate As Date -- The GMT value represents a valid date that has already been converted to GMT. DaylightAdjustment As DaylightAdjustmentKind (see "TzDateFallsWithinDaylightTime" on page 69 for more information about using DaylightAdjustmentKind).
Returns	A Date value
Related Topics	N/A

TzGMTToLocalAsString

Exposed In	Version 6.2
Function	This method is used to convert the GMTDate into the local date for the current time zone. The date is adjusted according to the rules of the DaylightAdjustment parameter and formatted according to the rules of the FormatString.
Syntax	Application.BasicFunctions.TzGMTToLocalAsString()

Parameters	GMTDate As Date - The GMT value represents a valid date that has already been converted to GMT. FormatString As String - A value used to format the date returned. See the Architect Help topic "Date/Time Field Format Strings" for more information. DaylightAdjustment As DaylightAdjustmentKind (see "TzGetDaylightName" on page 72 for more information about using DaylightAdjustmentKind).
Returns	String
Related Topics	N/A

TzGMTToLocalEx

Exposed In	Version 6.2
Function	This method is used to convert the GMTDate into a local date for the time zone represented by the TimeZoneKey. The result is adjusted according to the rules of the DaylightAdjustment parameter.
Syntax	Application.BasicFunctions.TzGMTToLocalEx()
Parameters	GMTDate As Date - The GMT value represents a valid date that has already been converted to GMT. TimeZoneKey As String -- The time zone identifier. For example: "Mexico Standard Time". DaylightAdjustment As DaylightAdjustmentKind (see "TzDateFallsWithinDaylightTime" on page 69 for more information about using DaylightAdjustmentKind).
Returns	A date value.
Related Topics	N/A

TzGMTToLocalExAsString

Exposed In	Version 6.2
Function	This method is used to convert the GMTDate into a local date for the time zone represented by the TimeZoneKey. The result is adjusted according to the rules of the DaylightAdjustment parameter and formatted according to the rules of the FormatString.
Syntax	Application.BasicFunctions.TzGMTToLocalExAsString()
Parameters	GMTDate As Date - The GMT value represents a valid date that has already been converted to GMT. TimeZoneKey As String -- The time zone identifier. For example: "Mexico Standard Time". FormatString As String - A value used to format the date returned. See the Architect Help topic "Date/Time Field Format Strings" for more information. DaylightAdjustment As DaylightAdjustmentKind (see "TzDateFallsWithinDaylightTime" on page 69 for more information about using DaylightAdjustmentKind).
Returns	String
Related Topics	N/A

TzIsValidTimeZoneKey

Exposed In	Version 6.2
Function	This method is used to determine whether or not the Value parameter is a valid TimeZoneKey for the local system.
Syntax	Application.BasicFunctions.TzIsValidTimeZoneKey()
Parameters	Value As String
Returns	Boolean
Related Topics	N/A

TzLocalToGMT

Exposed In	Version 6.2
Function	This method is used to convert the LocalDate into a GMT date for the current time zone. The result is adjusted according to the rules of the DaylightAdjustment parameter.
Syntax	Application.BasicFunctions.TzLocalToGMT()
Parameters	LocalDate As Date - A date that is local fo the time zone represented by the TimeZoneKey. DaylightAdjustment As DaylightAdjustmentKind (see "TzDateFallsWithinDaylightTime" on page 69 for more information about using DaylightAdjustmentKind).
Returns	A Date value
Related Topics	N/A

TzLocalToGMTAsString

Exposed In	Version 6.2
Function	This method is used to convert the LocalDate into a GMT date for the current time zone. The date is adjusted according to the rules of the DaylightAdjustment parameter and formatted according to the rules of the FormatString (see Architect Help).
Syntax	Application.BasicFunctions.TzLocalToGMTAsString()
Parameters	LocalDate As Date - A date that is local for the time zone represented by the TimeZoneKey. FormatString As String - A value used to format the date returned. See the Architect Help topic "Date/Time Field Format Strings" for more information. DaylightAdjustment As DaylightAdjustmentKind (see "TzDateFallsWithinDaylightTime" on page 69 for more information about using DaylightAdjustmentKind).
Returns	String
Related Topics	N/A

TzLocalToGMTEx

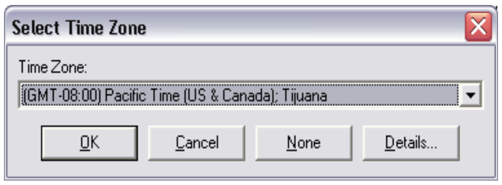
Exposed In	Version 6.2
Function	Thismethod is used to convert the LocalDate into a GMT date for the time zone represented by the TimeZoneKey. The result is adjusted according to the rules of the DaylightAdjustment parameter.
Syntax	Application.BasicFunctions.TzLocalToGMTEx()
Parameters	LocalDate As Date - A date that is local for the time zone represented by the TimeZoneKey. TimeZoneKey As String -- The time zone identifier. For example: "Mexico Standard Time". DaylightAdjustment As DaylightAdjustmentKind (for more information about using DaylightAdjustmentKind, see "TzDateFallsWithinDaylightTime" on page 69).
Returns	A Date value
Related Topics	N/A

TzLocalToGMTExAsString

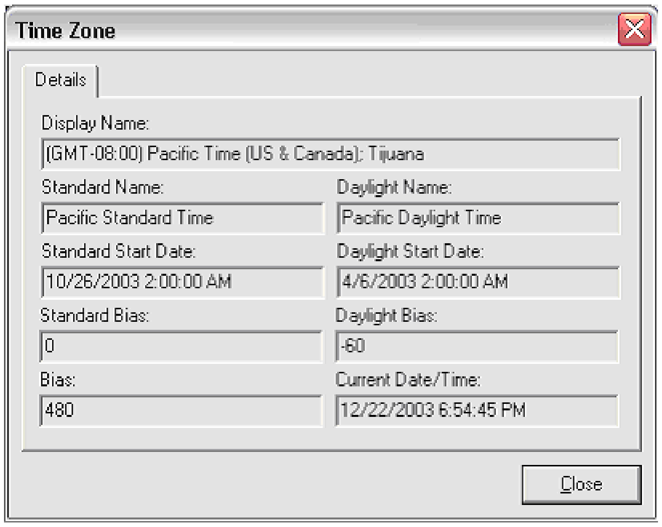
Exposed In	Version 6.2
Function	This method is used to convert the LocalDate into a GMT date for the time zone represented by the TimeZoneKey. The date is adjusted according to the rules of the DaylightAdjustment parameter and formatted according to the rules of the FormatString.
Syntax	Application.BasicFunctions.TzLocalToGMTExAsString()
Parameters	LocalDate As Date - A date that is local to the time zone represented by the TimeZoneKey. TimeZoneKey As String -- The time zone identifier. For example: "Mexico Standard Time". FormatString As String - A value used to format the date returned. See the Architect Help topic "Date/Time Field Format Strings" for more information. DaylightAdjustment As DaylightAdjustmentKind (see "TzDateFallsWithinDaylightTime" on page 69 for more information about using DaylightAdjustmentKind).
Returns	String
Related Topics	N/A

TzSelectTimeZone

Exposed In	Version 6.2
Function	This method displays the "Select Time Zone" dialog, allowing the user to select a time zone and to view the time zone's details:



Time Zone: (GMT-08:00) Pacific Time (US & Canada); Tijuana


Time Zone

Details

Display Name: (GMT-08:00) Pacific Time (US & Canada); Tijuana

Standard Name: Pacific Standard Time

Daylight Name: Pacific Daylight Time

Standard Start Date: 10/26/2003 2:00:00 AM

Daylight Start Date: 4/6/2003 2:00:00 AM

Standard Bias: 0

Daylight Bias: -60

Bias: 480

Current Date/Time: 12/22/2003 6:54:45 PM

Close

If the TimeZoneKey is defined, the dialog displays with the corresponding time zone already . If the user clicks OK, TzSelectTimeZone returns True, and the DisplayName parameter contains the time zone display name (e.g. "(GMT-07:00) Arizona") and the TimeZoneKey parameter contains the time zone key (e.g. "US Mountain Standard Time"). The time zone information is loaded from the user's registry from the location HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Time Zones (the TimeZoneKey represents a subkey of this key). The "World Time Zones" pick list is deprecated as of Sage SalesLogix version 6.2.

Syntax	Application.BasicFunctions.TzSelectTimeZone()
Parameters	TimeZoneKey As String -- The time zone identifier. For example: "Mexico Standard Time".
	DisplayName As Variant
	TimeZoneKey As Variant
Returns	Boolean
Related Topics	N/A

Example

```
Dim strDisplayName
Dim strTimeZoneKey
If Application.BasicFunctions.TzSelectTimeZone("US Mountain Standard Time",
strDisplayName, strTimeZoneKey) Then
If strTimeZoneKey <> "" Then
  MsgBox "The user : " & strDisplayName & " = """ & strTimeZoneKey & """"
Else
  MsgBox "The user  None."
End If
Else
MsgBox "The user canceled the dialog."
End If
```

TzTimeZoneObservesDaylightTime

Exposed In	Version 6.2
Function	This method is used to determine if the time zone, represented by the TimeZoneKey, observes Daylight Saving Time (see http://webexhibits.org/daylightsaving/index.html).
Syntax	Application.BasicFunctions.TzTimeZoneObservesDaylightTime()
Parameters	TimeZoneKey As String -- The time zone identifier. For example: "Mexico Standard Time".
Returns	Boolean
Related Topics	N/A
Example	The following code determines if the local time zone is currently observing Daylight Saving Time:

```
Const daForceAdjustment = 2
Dim strTimeZoneKey
Dim bIsDaylight
strTimeZoneKey = Application.BasicFunctions.TzGetLocalSystemTimeZoneKey
If Application.BasicFunctions.TzTimeZoneObservesDaylightTime(strTimeZoneKey) Then
  bIsDaylight = Application.BasicFunctions.TzDateFallsWithinDaylightTime(Now,
strTimeZoneKey, daForceAdjustment)
MsgBox "Daylight Saving Time = " & bIsDaylight
Else
MsgBox "The current time zone does *not* observe Daylight Saving Time."
End If
```

TzTranslateTimeZoneKey

Exposed In	Version 6.2
Function	This method is used to translate a TimeZoneKey for example, "US Mountain Standard Time") into a time zone display name (for example, "(GMT-07:00) Arizona").
Syntax	Application.BasicFunctions.TzTranslateTimeZoneKey
Parameters	TimeZoneKey As String -- The time zone identifier. For example: "Mexico Standard Time".
Returns	String
Related Topics	N/A
Example	See "TzDateFallsWithinDaylightTime" on page 69.

UpdateFileAttachment

Function	Provides a means to send an attachment update to Remotes.
Object	Application.BasicFunctions.UpdateFileAttachment
Syntax	UpdateFileAttachment()
Parameters	AttachID as String (path)
Returns	Boolean
Related Topics	N/A

UpgradeWordTemplate

This function is deprecated in version 6.2.

Exposed In	Version 5.2	
Function	Used to upgrade Mail Merge templates from 4.x to 5.x.	
Object	Application.BasicFunctions.UpgradeWordTemplate	
Syntax	UpgradeWordTemplate()	
Parameters	FileName as string	The Word template filename that is to be upgraded. This is a full file system path to the template on the user's local machine.
	PluginName as string	The name to use for the new plugin record.
	PluginDescription as string	The description to use for the new plugin record.
	aPluginFamily as string	The family to use for the new plugin record.
Returns	String	
Related Topics	N/A	

WebEncrypt

Exposed In	Version 6.2.1
Function	Returns an encrypted value for the string passed in.
Object	Application.BasicFunctions.WebEncrypt
Syntax	WebEncrypt
Parameters	Value as String
Returns	String
Related Topics	N/A

WebOpen

Function	Opens the specified document with the user’s default World Wide Web browser.
Object	Application.BasicFunctions.WebOpen
Syntax	WebOpen *path*
Parameters	URL as String (path)
Returns	Boolean
Related Topics	N/A

Application.BringToFront

Exposed In	Version 6.2
Function	When several windows are open, this function brings Sage SalesLogix to the front.
Object	Application.BringToFront
Syntax	BringToFront
Parameters	None
Returns	Boolean
Related Topics	N/A

Application.Clipboard

AsText

Exposed In	Version 6.2
Function	Returns the current contents of the Windows clipboard.
Object	Application.Clipboard.AsText
Syntax	AsText(= Value)
Parameters	None
Returns	String - the current contents of the clipboard.
Related Topics	N/A

Clear

Exposed In	Version 6.2
Function	Clears the Windows clipboard.
Object	Application.Clipboard.Clear
Syntax	Clear
Parameters	None
Returns	Boolean
Related Topics	N/A

Application.ConnectionString

Exposed In	Version 6.2
Function	Returns the current ADO connection string.
Object	Application.ConnectionString
Syntax	ConnectionString
Parameters	None
Returns	The current connection string being used by the Client
Related Topics	N/A

Application.CreateObject

Function	Used to open a recordset from an external application. Use SLXApplication.CreateObject ("ADODB.Recordset") instead of using CreateObject("ADODB.recordset"). the ADO.Connection object returned by SLXApplication.GetNewConnection and the ADODB.Recordset object are created in the same address space (saleslogix.exe). This is a requirement for Recordset.Open to work properly.
Object	Application.CreateObject
Syntax	CreateObject
Parameters	None
Returns	Object Value
Related Topics	N/A

Application.Debug

Assert

Exposed In	Version 6.2
Function	Returns the specified error (message) if the condition evaluates to False.
Object	Application.Debug.Assert
Syntax	Assert
Parameters	Condition as Boolean Message as String - the error message to display.
Returns	None
Related Topics	N/A

Example

```
Condition = (cmbTest.Items.Count > 0)
Application.Debug.Assert condition, "assertion error: the number of items must be greater
than 0"
```

Fail

Exposed In	Version 6.2
Function	Returns the specified error (message).
Object	Application.Debug.Fail
Syntax	Fail
Parameters	Message as String - the error message to display.
Returns	None
Related Topics	N/A

Example

```
Application.Debug.fail "testing fail"
```

WriteLine

Exposed In	Version 6.2
Function	Writes a message to the event log using the Win32 API function OutPutDebugString.
Object	Application.Debug.WriteLine
Syntax	WriteLIne()
Parameters	Message as String - the error message to display.
Returns	None
Related Topics	N/A

Example

```
Application.Debug.WriteLine "testing writeline"
```

WriteLineIf

Exposed In	Version 6.2
Function	Writes a message to the event log based on a condition. Uses the Win32 API function OutPutDebugString if Condition evaluates to True.
Object	Application.Debug.WriteLineIf
Syntax	WriteLineIf()

Parameters	Condition as Boolean
	Message as String
Returns	None
Related Topics	N/A

Example

```
Condition = (cmbTest.Items.Count = 0)
Application.Debug.WriteLineIf condition, "the number of items is 0"
```

Application.DoEvents

Exposed In	Version 7.0
Function	In lengthy operations, allows the application to respond to messages.
Object	Application.DoEvents
Syntax	DoEvents
Parameters	None
Returns	Boolean
Related Topics	N/A

Application.Environment

Exposed In	Version 7.0
Function	Use to resolve alignment issues when display settings are not 96DPI.
Object	Application.Environment
Syntax	Environment()

Parameters	DesktopHeight	Specifies the height of the entire virtual desktop.
	DesktopLeft	Specifies the x-coordinate of the desktop’s left edge relative to the upper-left corner of the primary monitor.
	DesktopTop	Specifies the y-coordinate of the entire destop’s top edge relative to the upper-left corner of the primary monitor.
	DesktopWidth	Specifies the width of the entire virtual desktop.
	Height	Indicates the vertical size of the screen in pixels.
	PixelsPerInch	Indicates the number of screen pixels that make up a logical inch in the vertical direction.
	ThemesEnabled	Indicates if Windows XP themes are enabled. Returns True if Themes are enabled. Returns False if Themes are not enabled.
	Width	Indicates the horizontal size of the screen in pixels.

Returns	None
Related Topics	N/A

Example

```
Control.Top = cLng(Application.Environment.PixelsPerInch * Control.Top / 96)
```

Application.Forms

This object contains a collection of a Sage SalesLogix Form. This can be useful to reference a displayed Form in script.

Example To set the 'Assistant' edit box on the "Contact Details" view, use the following:

```
Set Form = Application.Forms("System:Contact Detail")
Form.txtAssistant.Text = "John Doe"
```

Forms Collection

Properties	Description
Count	Returns the number of the currently loaded forms.
Item	A simple wrapper around the Tab exposing the Visible property and a reference to the Active Form it contains.
ActiveTabIndex	Use to Set or Get the active tab.

Count

Function	Read-only Returns the number of the currently loaded forms.
Object	Application.Forms.Count
Syntax	Count (= Value)
Parameters	None
Returns	Integer
Related Topics	N/A

Item

Function	Read/write. Returns the specified form as a generic object. Index can be either an integer or a string (plugin ID or plugin name).
Object	Application.Forms.Item (Index)
Syntax	Item(Index as Variant) [= value]
Parameters	Variant - the Plugin name or an index.
Returns	The form object.
Related Topics	N/A

Application.GetNewConnection

Function	Returns a new initialized instance of the ADO Connection object to the current database.
Object	Application.GetNewConnection
Syntax	GetNewConnection
Parameters	None
Returns	Object Value
Related Topics	N/A

Application.GlobalInfo Object

This object is used for sharing values between views and scripts. Use this object to interact with Global variables.

Legacy functions and corresponding methods in the Application.BasicFunctions (such as GlobalInfoFor and GlobalInfoSet) will still function and use the same Global variables.

GlobalInfo Collection

Properties	Description
Add	Takes two parameters - name (string - the name being used to call your global object) and value (variant - any data type including a COM object and returns an integer (the index in the collection. First added = 0, second added = 1 and so forth.
Count	The number of items in the collection.
Delete	Parameter - Index, an OLE varient (might be an integer, or possibly the name of the object you want to delete.)
IndexOf	Function Parameter - name - string Returns - the position of the item number in the collection.
Item	Index parameter - variant - either the name or the index. Returns - the object itself or the value for that particular global info object.

CTRL-F5 destroys the objects in the GlobalInfo objects collection.

Add

Function	Adds a global variable with the specified name and value to the GlobalInfo collection. If an object with the same name already exists, it is overwritten with the new value. The method returns an integer index of the object.
Object	Application.GlobalInfo.Add
Syntax	Add (Name, Value)
Parameters	Name as String. The name to be given to your global object. Value as Variant - the value of the variable to be added. This can be any data type, including a COM object.
Returns	A Long Value
Related Topics	N/A

Count

Function	Read-only. Returns the number of named values in the GlobalInfo object.
Object	Application.GlobalInfo.Count
Syntax	Count

Parameters	Value as Long. The value to be counted.
Returns	Long
Related Topics	N/A

Delete

Function	Deletes the specified global object.
Object	Application.GlobalInfo.Delete
Syntax	Delete
Parameters	Index as Variant. The index can be either the name of an object or an integer.
Returns	Boolean
Related Topics	N/A

IndexOf

Function	Returns the index of an object with a given name or -1 if the object is not found.
Object	Application.GlobalInfo.IndexOf
Syntax	IndexOf
Parameters	Name as String. The name of the object for which the index is being returned.
Returns	Long
Related Topics	N/A

Item

Function	Read/write. Returns/sets a global variable.
Object	Application.GlobalInfo.Item
Syntax	Item
Parameters	Value as Variant. Index can be either a string (name of the global object) or an integer.
Returns	Returns/sets a global variable.
Related Topics	N/A
Example	The following examples set a global variable named “MyVariable” to “Test Value”.

```
Application.GlobalInfo.Add(“MyVariable”, “TestValue”)
Application.GlobalInfo.Item(“MyVariable”)=“TestValue”
Application.GlobalInfo(“MyVariable”)=“TestValue”
Application.GlobalInfo.MyVariable = “TestValue”
Application.BasicFunctions.GlobalInfoSet “MyVariable”, “Test Value”
```

Application.MainViews

MainView Object (IMainView)

Design Time: set properties using the property box.

Runtime:

```
Dim objMV
Set objMV = Application.MainViews.({example}AddEx)
```

Properties	**Description**
BaseTable	Read-only. Returns the strTableName.

BorderStyle	Uses the following enumeration for possible values:
	bsNone = 0
	bsSingle = 1
	bsSizeable = 2
	bsDialog = 3
	bsToolWindow = 4
	bsSizeToolWindow = 5
	This property is only available at run-time when the Main view is called from script.

Caption	String that appears in the caption bar to show the title of the window. This property is available at both design time and run time.
CaptionBarVisible	Caption Bar is the bar at the top of the window. It contains the First, Previous, Next, Last, and QuickFind buttons. This property has a Boolean value and the default is False.
CurrentGroupSQL	The SQL of the Current Group.
CurrentID	Sage SalesLogix ID String. The KeyField ID of the base table for the Main view. Automatically updates the Main view to display the record attached to the KeyField ID. Available at run-time only.
DataMode	A read-only property that returns the DataMode (see SetDataMode).
DefaultLookup	Not in use. Use QuickFindLookup.
DefaultTabs	Design-time only - applies to the Tabs pane. Provides access to set the default display order for the tabs.
DetailsView	A Read-only reference to the Active Form contained within the Details View pane. Returns the form object contained in this pane. < Family:Name > for example, "Account:AccountDetail"
DisplayMode	Variant: mvmDetails (1) shows the details view mvmList (2) shows the list mode (groups) with the item. mvmSplit (3) shows the Split view. True opens last mode used to display. False defaults to the List View.
EntityNamePlural	Controls the name used for the QuickFind button when there are multiple records to locate. Available at design time only.
EntityNameSingular	Controls the name used for the QuickFind button when there is only one record to locate. Available at design time only.
GroupID	String. This is the Sage SalesLogix group ID for the record that you want to display. Passing an empty ID loads the first record in the current group.

Properties	Description
GroupsPane	Provides access to the collection of keyfields. Read-only. Popup menu available at design time and run time.
GroupsPaneVisible	Boolean. Available at run time. This property is comparable to the Visible property available to the Groups Pane on the MainView through the Architect.
Height	Sets the height of the Active form.
Left	Sets the location for the left edge of the Active form.
MiddleView	A read-only reference to the Active Form contained within the Middle View pane. Returns the form object contained in this pane. MiddleView can be set at run time using the same format as TabsView: `objMainView.MiddleView = "Opportunity:Notes-History"`
MiddleViewVisible	Boolean. Available at run time. This property is comparable to the Visible property available to the MainView through the Architect.
MoreTabsVisible	Boolean. Available at runtime as read-only.
Name	Returns the name of the MainView as set at design-time.
PopupMenu	MainView form. Available at design-time through the popup editor. Available at runtime through the IPopup Menu Interface: IPopup Menu interface • procedure Popup(X: Integer; Y: Integer); fires the popupmenu in the designated location. • property AutoHotKey: Boolean. • property AutoLineReduction: Boolean. • property Handle: Integer (read-only). • property Items: IMenuItemX (read-only). IMenuItemX interface • property Checked: Boolean • property Caption: String • property Count: Integer. (read-only) • propertyEnabled: Boolean • property Handle - OLE_HANDLE (read-only) • property Items [Index:Integer]: IMenuItemX (read -only) • property MenuIndex: Integer • property Parent: IMenuItemX • property RadioItem: Boolean • property Visible: Boolean • function Add: IMenuItemX • procedure Clear • procedure Click • procedure Delete(Index: Integer) • function Find(const Caption: String):IMenuItemX • function Insert(Index: Integer): IMenuItemX • function IsLine: Boolean • property Default: Boolean

Properties	**Description**
QuickFindLookup	Default lookup for QuickFind for the current MainView. Not available at run time.
QuickFindShowMore	Boolean to hide or show the More button on the QuickFind. Default is False. Not available at run time.
QuickFindVisible	Boolean to show or hide the QuickFind button. Not available at run time.
Tabs Collection	Count: Returns the number of tabs. Item(Index): Either an integer (O to Count-1) or a string. Given an index, returns a Tab object. In the case of a string, Index is either the plugin name (in the Family:Name form) or the plugin ID. Add(ID) Adds and makes active a plugin with a given ID or name (Family:Name). Remove(ID): Closes a tab that displays the form with a given name or ID. ActiveTab (r/w): Returns or sets the active tab (Tab object.) ActiveTabIndex (r/w, integer): Returns or sets the active tab by its index. Height (r/w, integer): Returns or sets the height of the tabs area. Tab Object Delete: Closes the tab. Caption (r/w, string) Returns or sets the tab's caption. Form: Returns the Form object (this object is also accessible from the form's script). The following example returns the text of the form's control named Edit1: Form.Edit1.Text Index: An integer index of the tab Kind: tkUnknown = $00000000 tkLegacyForm = $00000001 tkActiveXForm = $00000002 tkMoreTabs = $00000003 tk= $00000004 tkActivities = $00000005 tkAssociations = $00000006 tkProcesses = $00000007 tkSummary = $00000008

Properties	Description
TabsView	Container for the Tabs. Tabs available for this view depend on the base table to be the MainTable for the MainView. At runtime, this property returns a reference to the active tab view. For example, to set the active tab view `:MainView.TabsView = (Family:tab(where tab is the plugin name))` The Administrator can override the tabs that are visible. (In the Administrator, go to Tools>Options>Client Plugins>Main View Tabs.) **Note:** Although TabsView is still supported in Sage SalesLogix versions 6.2.2 and later, it has been replaced by the Tabs Collection.
TabsViewCount	Shows the user the number of tabs available. Does not include MoreTabs.
TabsViewVisible	Boolean. Available at run time. This property is comparable to the Visible property available to the Tabs pane on the MainView through the Architect.
Top	Sets the location for the top of the Active form.
TabsViews	Provides the reference to the TabsView panel pane.
Visible	Boolean. Specifies whether the user can see that section of the form. Available at design. See MiddleViewVisible property for availablility at run-time.
ViewName	DetailsView and MiddleView both have a ViewName property.
Width	Sets the width of the Active form. You must set this property when the Main View is a wrapper around an Active form.

Methods	Description
BringToFront	Brings the Main View to the front.
Close	Closes the Main View window.
GoToDefaultRecord	Goes to the first record of the group when there is a group available.
Hide	Hides the Main View window.
Maximize	Expands the Main View window to available dimensions.
Minimize	Minimizes the Main View window.
Modified	Flags that data in the Main View has changed.
Post	Updates / inserts data.
Refresh	Reloads the Main View to show updated data.
RestoreSize	Expands the window to the last size used by the user. Pulls the information from UserOptions.
SaveSize	Saves the setting for the current size of the window.

Methods	Description
SetDataMode	There are two modes: Insert: the data driver assumes the record does not exist and creates a new one. Edit: the data driver assumes the record does exist and updates it. This is the default setting.
Show	Shows the Main View non-modally.
ShowIDAsLookupResult	Passes a comma-delimited list of Key Field IDs and creates a lookup results group.
ShowModal	Shows the Main View modally.

Creating a Main View at Runtime

It is possible to create a modal Main View (for example, the Insert dialog box in the Sage SalesLogix Client). If you pass the name of a form (not a Main View) to Application.MainViews.Add(), Sage SalesLogix assumes that you want to display a main view-style container with only the details area visible. It creates an instance of the MainView object dynamically (that is, it is not defined in the database), sets the MainView details pane name to the value passed in, and hides all other areas (toolbar, tabs and so forth.) The following example shows how to display the Ticket:Details active form modally:

```
SetMV = Application.MainViews.Add("Ticket:Details:, 0, FALSE)
MV.CurrentID = <ticket record ID>
If MV>showModal = 1, then 'OK was clicked
' specify action here
End If.
```

To specify buttons that can close a modally displayed view, add buttons to the form using the ModalResult property set to the desired value (for example, mrOK.) If the view is non-modal, these buttons will not function.

ActiveView

Exposed In	Version 6.2
Function	Returns the currently active Main View.
Object	Application.MainViews.ActiveView
Syntax	ActiveView
Parameters	Value - a value in the Main View object. See "MainView Object (IMainView)" on page 88 for more information.
Returns	IMainView or MainView object

Add

Exposed In	Version 6.2
Function	Displays a Main View. Adds the Main View object to the MainView Collection.
Object	Application.MainViews.Add

Syntax	Add()	
Parameters	View name as String ()	View name is the Family:Name, for example, "Personal:Ticket Details". This can be either a MainView plugin or a Form plugin, but not a legacy Basic form.
	WindowStyle	One of the TxMainViewStyle enumeration values: mvsDefault (0) Displays the Main View in a regular MDI window. mvsMDIChild (1) the window is inside the Sage SalesLogix window. This is the most common WindowStyle used in Sage SalesLogix. mvsStayOnTop (2) the window is always on top. When minimized, the window is outside the Sage SalesLogix window. For example, the AccountTickets tab.
	OpenExisting as Boolean	Specifying False always creates a new instance of the particular MainView. Specifying True forces Sage SalesLogix to return an existing Main View if a Main View with the same name (Family:Name) is already open. Note: This property does not apply to forms, only Main Views.
Returns	A value in the MainView object and adds the MainView object to the MainView Collection.	
Related Topics	"Creating a Main View at Runtime" on page 93	
Example	The following example opens a new "Personal:Ticket Details" Main View, displays it, and then displays a Ticket record:	

```
Set MV = Application.MainViews.Add("Personal:Ticket Details", 1, TRUE)
MV.Show
MV.CurrentID = <some ticket record id>
```

AddEx

Exposed In	Version 6.2	
Function	Passing parameters to set the additional parameters.	
Object	Application.MainViews.AddEx	
Syntax	AddEx	
Parameters	View name as String ()	View name is the Family:Name, for example, "Personal:Ticket Details". This can be either a MainView plugin or a Form plugin, but not a legacy basic form.
	WindowStyle	One of the TxMainViewStyle enumeration values: mvsDefault (0) Displays the Main View in a regular MDI window. mvsMDIChild (1) the window is inside the Sage SalesLogix window. This is the most common WindowStyle used in Sage SalesLogix. mvsStayOnTop (2) the window is always topmost. When minimized, the window is outside the Sage SalesLogix window. For example, the AccountTickets tab.

OpenExisting as Boolean	Specifying False always creates a new instance of the particular Main View. Specifying True forces Sage SalesLogix to return an existing MainView if a Main View with the same name (Family:Name) is already open. Note: This property does not apply to forms, only Main Views.
DisplayMode as Variant	mvmDetails (1) - shows the Details view. mvmList (2) - shows the list mode (groups) with the item . mvmSplit (3) - shows the Split view. True - remembers the last DisplayMode state for that Main View and displays in that mode. False - Ignored. Defaults to the list view.
RecordID as String	The Sage SalesLogix record ID for the record that you want to display. For example, if you want to display John Adams, then you would type in his record ID. Passing an empty ID loads the first record for the GroupID passed.
GroupID as String	The Sage SalesLogix group ID for the record that you want to display. Sets that group as the current group regardless of whether or not the RecordID is within that group. Passing an empty GroupID results in it using the Current Group. Passing an empty RecordID loads the first Record for the GroupID passed. If the GroupID and the RecordID are empty, the first record in the current group is displayed.

Returns A value in the MainView object and adds the MainView object to the MainView collection. If a detail form is called, it is added to the Form collection.

Related Topics “Creating a Main View at Runtime” on page 93

Count

Exposed In Version 6.2

Function Displays the number of open Main Views.

Object Application.MainViews.Count

Syntax Count

Parameters None

Returns An integer representing the number of open Main Views in the Main View collection.

Related Topics N/A

GetViewForRecord

Exposed In	Version 6.2
Function	Use to create or reuse a default Main View for a particular table and display the specified record.
Object	Application.MainView.GetViewForRecord
Syntax	GetViewForRecord
Parameters	ID as String - the KeyField ID.
	Table as String - the Main Table.
Returns	A value from the IMainView collection. It displays a record for the Main Table passed for the specified ID.
	If there is more than one Main View defined for that Main Table, the Main View displayed is the first one located. There is no guarantee which table that will be.
Related Topics	N/A

Item

Exposed In	Version 6.2
Function	Returns an existing Main View with a given index (0 through Count - 1).
Object	Application.MainViews.Item
Syntax	Item(Index as Variant) [= value]
Parameters	Value - an IMainView object
Returns	An IMainView object reference.
Related Topics	N/A

Application.Managed

Create

Exposed In	Version 7.0
Function	Creates a new instance of a .NET Form.
Object	Application.Managed.Create
Syntax	Create()
Parameters	Title - a .NET form title (string)
	ClassName - a .NET form class name (string)
Returns	String - a new form handle.
Related Topics	N/A

Destroy

Exposed In	Version 7.0
Function	Destroys a specified instance of a .NET Form.
Object	Application.Managed.Destroy
Syntax	Destroy()
Parameters	Handle - string returned by Create method.

Returns	0 - instance and application domain successfully destroyed.
	-1 - instance not found.
	1 - instance successfully destroyed but domain not unloaded.
Related Topics	N/A

Run

Exposed In	Version 7.0
Function	Calls a specified instance of a .NET Form.
Object	Application.Managed.Run
Syntax	Run()
Parameters	Handle - string returned by Create method.
	Args - optioinal arguments (integer, string, safe array, etc.)
Returns	A result from Runnable.Run method.
Related Topics	N/A

Application.Name

Function	This property returns a string name for the Sage SalesLogix application running the script. For example, "SalesLogix" is the Sage SalesLogix Client.
Object	Application.Name
Syntax	Name
Parameters	Value as String
Returns	String
Related Topics	N/A

Application.PickLists

This object contains a collection of Sage SalesLogix pick lists.

Count

Function	Read-only. Returns the number of items in the collection of Sage SalesLogix pick lists.
Object	Application.PickLists.Count
Syntax	Count (= Value)
Parameters	Value as Long
Returns	Long
Related Topics	N/A

Item(Index)

Function	Read/write. Returns a PickList object with a specified index. Index valid range is 0 to Count -1.
Object	Application.PickLists.Item
Syntax	Item(Index As Variant) [= value]

Parameters A value from IPickList

Add	Adds an entry to the pick list.
AllowEdit	Boolean. Specifies whether a user can edit items in the pick list.
Count	Returns the number of items in the pick list.
DefaultItem	Returns the index of the default pick list item.
Fixed	Boolean. Specifies whether a user a string different than one of the pick list items in an edit box associated with the pick list.
ID	Returns the ID of the pick list.
Item(Index)	Returns an instance of a PickListItem object. Index valid range is 0 to Count-1.
ItemBySequence	The pick list item index.
ItemByShortText(Value)	Returns a pick list item (see PickListItem object) matching the specified parameter.
ItemByText(Value)	Returns a pick list item (see PickListItem object) matching the specified parameter.
MultiSelect	Boolean. Specifies whether a user can select multiple items from the list.
Name	Returns the name of the pick list.
RequiredEntry	Boolean. Specifies whether an entry from the list must always be .
Save	Commits any changes made to the pick list during this session to the database.
Select	Launches the pick list with an item pre-. Columns determine what pick list columns are displayed. Values are: 1 - Orders 2- Codes 4 - Items
Sorted	Boolean. Specifies whether the values displayed to the user are sorted

Returns None

Related Topics N/A

Manage

Function	Displays a dialog box used to manage the Sage SalesLogix pick lists.
Object	Application.PickLists.Manage
Syntax	Manage
Parameters	None
Returns	None
Related Topics	N/A

Select

Function	Displays a dialog box with the contents of the pick list and returns a collection of items (see Collection object). If a user cancels a dialog, NULL is returned.
Object	Application.PickLists.Select
Syntax	Select
Parameters	None
Returns	String
Related Topics	N/A

Application.Quit

Exposed In	Version 6.2
Function	Shuts down Sage SalesLogix.
Object	Application.Quit
Syntax	Quit
Parameters	None
Returns	Boolean
Related Topics	N/A

Application.Reports

Count

Exposed In	Version 7.0
Function	Returns the number of currently loaded reports.
Object	Application.Reports.Count
Syntax	Count
Parameters	None
Returns	Integer
Related Topics	N/A

Item (Index)

Exposed In	Version 7.0
Function	Returns an IReport.
Object	Application.Reports.Item
Syntax	Item(Index As OleVariant) [= value]
Parameters	A value from IReports.

DateField	Master Date Field as defined in the Report Properties dialog (used for filtering).
FilterConditions	Add filter conditions to a specific report at design time.
GetRecordSelectionFormulaForGroup	Gets Crystal Syntax formatted RecordSelectionFormula for any Sage SalesLogix Group.

MainTable	Main Table as defined in the Report Properties dialog (used for filtering).
NativeObject	Reference to Crystal RDC object.
PluginID	PluginID of the report property.
UseDateFilter	Show Date Filter as defined in the Report Properties dialog.
UseGroupFilter	Show Group Filter as defined in the Report Properties dialog.
UserField	Master User Field as defined in the Report Properties dialog (used for filtering).
UseUserFilter	Show User Filter as defined in the Report Properties dialog.

Returns OleVariant

Related Topics N/A

Example

```
'get instance of Account Summary IReport
Dim objRoport
Set objReport = Application.Reports.Item("Account Summary - Sample")
```

Application.ShowActivityListWindow

Exposed In Version 7.5

Function If not already open, opens the Activities List Main view and makes it active.

Object SLXApplication.ShowActivityListWindow

Also see "ActivityListWindow Object" on page 131.

Syntax ShowActivityListWindow

Parameters None

Returns N/A

Application.State

Exposed In Version 6.2

Function Represents the current state of the application.

Object SLXApplication.State

Syntax State

Parameters Value - an ApplicationState value

Constant	Value
asLoading	0
asReady	1
asQuitting	2
asFullRefresh	3

Returns None

Application.Translator

Charset

Exposed In	Version 6.2.1
Function	Returns the character set of the font that is used by the translator when performing the translation.
Object	Application.Translator.Charset
Syntax	Currency.Charset
Parameters	None
Returns	Integer. The current character set obtained from windows regional settings.
Related Topics	N/A

CurrencyDecimals

Exposed In	Version 6.2
Function	Returns the number of digits to the right of the decimal point in a currency amount.
Object	Application.Translator.CurrencyDecimals
Syntax	Currency.Decimals
Parameters	None
Returns	Byte. CurrencyDecimals is the number of digits to the right of the decimal point in a currency amount.
Related Topics	N/A

CurrencyFormat

Exposed In	Version 6.2
Function	Defines the currency symbol placement and separation. Possible values are: 0 = $1 ; 1 = 1$; 2 = $ 1 ; 3 = 1 $
Object	Application.Translator.CurrencyFormat
Syntax	CurrencyFormat
Parameters	None
Returns	Byte
Related Topics	N/A

CurrencyString

Exposed In	Version 6.2
Function	Defines the currency symbol (or characters).
Object	Application.Translator.CurrencyString
Syntax	CurrencyString
Parameters	None
Returns	String
Related Topics	N/A

DateSeparator

Exposed In	Version 6.2
Function	DateSeparator is the character used to separate the year, month, and day parts of a date value.
Object	Application.Translator.DateSeparator
Syntax	DateSeparator
Parameters	None
Returns	Char
Related Topics	N/A

DecimalSeparator

Exposed In	Version 6.2
Function	DecimalSeparator is the character used to separate the integer part from the fractional part of a number.
Object	Application.Translator.DecimalSeparator
Syntax	DecimalSeparator
Parameters	None
Returns	Char
Related Topics	N/A

LocalDecimalToUSDecimal

Not needed or supported in VBScript. VBScript internally handles the numeric issues associated with regional settings.

When using VBScript and ADO, Sage SalesLogix highly recommends parameterized queries which eliminate the use of an Update statement.

Exposed In	Version 6.2
Function	Used to convert a local operating system value (other than US) to a US value, for use in code. Data values from controls in the Sage SalesLogix Client are passed to the scripting engine as strings. A string value formatted as a non-US standard number must to be converted to US standard number format before it can be used by the scripting engine. The converted value must then be converted to a valid number before it is subject to arithmetic or value compare functions.
Object	Application.Translator.LocalDecimalToUSDecimal
Syntax	LocalDecimalToUSDecimal
Parameters	Value as String
Returns	String
Related Topics	N/A

Example

```
Dim Value1
Dim ValueResult

  Value1 = Application.Translator.LocalDecimalToUSDecimal(textbox.text)  'textbox.text =
   1.234,56 for German Regional Settings
  ValueResult = Value1 * 3.14  'Calculations are done in common US Regional Settings
```

Localize

Exposed In	Version 6.2
Function	When you pass a string to this function, if that string exists in the Dictionary, the localized version of the string is returned. If the string to be translated cannot be found in the dictionary, Localize returns the string passed to it, untranslated.
Object	Application.Translator.Localize
Syntax	Localize
Parameters	Value as String
Returns	String
Related Topics	N/A

LocalString contains the translation for "An error has occurred". If this phrase is not in the dictionary, LocalString contains "An error has occurred" in English.

LocalizeFont

Exposed In	Version 6.2
Function	Changes the character set for the font property for third party controls in cases where Font property is not called Font.
Object	Application.Translator.LocalizeFont
Syntax	LocalizeFont
Parameters	Font as IFont. Refer to Microsoft Developers Network for information on IFont.
Returns	None
Related Topics	N/A

LongDateFormat

Exposed In	Version 6.2
Function	LongDateFormat is the format string used to convert a date value to a long string suitable for display but not for editing.
Object	Application.Translator.LongDateFormat
Syntax	LongDateFormat
Parameters	None
Returns	String suitable for display.
Related Topics	N/A

ShortDateFormat

Exposed In	Version 6.2
Function	ShortDateFormat is the format string used to convert a date value to a short string suitable for editing.
Object	Application.Translator.ShortDateFormat
Syntax	ShortDateFormat
Parameters	None
Returns	String suitable for editing.
Related Topics	N/A

StrToInt

Exposed In Version 7.5

Function	Converts a string to an integer. Unlike the built-in VB functions, which may not always correctly use the localized thousands separator, this function always uses the thousands separator specified in the Regional settings for the current Windows user.
Object	Application.Translator.StrToInt
Syntax	StrToInt
Parameters	String
Returns	Integer
Related	Topics Application.Translator.StrToFloat

StrToFloat

Exposed In Version 7.

Function	Converts a string to a float. Unlike the built-in VB functions, which may not always correctly use the localized thousands and decimal separators, this function always uses the thousands and decimal separators specified in the Regional settings for the current Windows user.
Object	Application.Translator.StrToFloat
Syntax	StrToFloat
Parameters	String
Returns	Float
Related Topics	Application.Translator.StrToInt

ThousandSeparator

Exposed In	Version 6.2
Function	ThousandSeparator is the character used to separate thousands in numbers with more than three digits to the left of the decimal separator.
Object	Application.Translator.ThousandSeparator
Syntax	ThousandSeparator
Parameters	None
Returns	String
Related Topics	N/A

TimeAMString

Exposed In	Version 6.2
Function	TimeAMString is the suffix string used for time values between 00:00 and 11:59 in 12-hour clock format.
Object	Application.Translator.TimeAMString
Syntax	TimeAMString
Parameters	None
Returns	String
Related Topics	N/A

TimePMString

Exposed In	Version 6.2
Function	TimePMString is the suffix string used for time values between 12:00 and 23:59 in 12-hour clock format.
Object	Application.Translator.TimePMString
Syntax	TimePMString
Parameters	None
Returns	String
Related Topics	N/A

TimeSeparator

Exposed In	Version 6.2
Function	TimeSeparator is the character used to separate the hour, minute, and second parts of a time value.
Object	Application.Translator.TimeSeparator
Syntax	TimeSeparator
Parameters	None
Related Topics	N/A
Returns	String

USDecimalToLocalDecimal

Not needed or supported in VBScript. VBScript internally handles the numeric issues associated with regional settings.

When using VBScript and ADO, Sage SalesLogix highly recommends parameterized queries which eliminate the use of an Update statement.

Exposed In	Version 6.2
Function	Having converted a local operating system value (other than US) to a US value, for use in code, this function is used to convert the result back to the regional setting of the operating system.
Object	Application.Translator.USDecimalToLocalDecimal
Syntax	USDecimalToLocalDecimal
Parameters	Value as String
Returns	String. The regional setting for the operating system.
Related Topics	N/A

Example

```
Dim Value1
Dim ValueResult

Value1 = .LocalDecimalToUSDecimal(textbox.text)  'textbox.text = 1.234,56 for German
Regional Settings
ValueResult = Value1 * 3.14  'Calculations are done in common US Regional Settings
Textbox.text = Application.Translator.USDecimalToLocalDecimal(ValueResult)  'Converts the
result back to German Regional Settings and displays the result in the textbox.
```

If Value1 is not converted to US Regional Settings, the calculation is incorrect or in the case of French RS, the calculation fails.

Application.UserOptions

This function gets an instantiated object being used by the system and caches user options as they are read to improve performance. All writes to options are immediate.

Many of these functions return errors if:

- The length of the OptionName or the OptionCategory exceeds the maximum allowed.
- A bad character is used.
- The OptionName or OptionCategory are not unique.

All SetAs and GetAs methods generate an exception error if the user option does not exist in the default table. In all cases where errors are returned, refer to the Microsoft VBScript Error Handling documentation.

Add

Exposed In	Version 6.2
Function	Adds a definition value for a User Option.
Object	Application.UserOptions.Add
Syntax	Add
Parameters	OptionName as String: The name of the option. In combination with OptionCategory, this provides the primary key and must be unique. Maximum length is 64 characters. OptionCategory as String: The Family of the option for categorization. In combination with OptionName, this provides the primary key and must be unique. It does not accept quotation marks as part of the name. Maximum length is 64 characters. OptionDisplayName as String: "Pretty" name. Maximum length is 64 characters. OptionValue as String: This is the value to be stored. In this function, this is the default value that applies to all users unless they have an overriding option of their own.
Returns	None
Related Topics	N/A

ClearCategory

Exposed In	Version 6.2
Function	Deletes all options in the user options tables for a given user for the category at the child level. Resets to the default value.
Object	Application.UserOptions.ClearCategory
Syntax	ClearCategory
Parameters	OptionCategory as String: The Family of the option for categorization. In combination with OptionName, this provides the primary key and must be unique. It does not accept quotation marks as part of the name. Maximum length is 64 characters.
Returns	None
Related Topics	N/A

ConnectionString

Exposed In	Version 6.2
Function	Prepopulated with the Connection string to the database for the currently logged in user. Used externally.
Object	Application.UserOptions.ConnectionString
Syntax	ConnectionString
Parameters	None
Returns	String. The Sage SalesLogix connection string to the database for the logged in user.
Related Topics	N/A

CopyAll

Exposed In	Version 6.2
Function	Copies all of the current user's options (the user specified in the UserID) to the destination user (DestUserID). This function is used in the Administrator to create a user template.
Object	Application.UserOptions.CopyAll
Syntax	CopyAll
Parameters	The UserID for the Administrator.
Returns	None
Related Topics	N/A

Exists

Exposed In	Version 6.2
Function	Determines if this user has an entry for this option. A default value in the useroptiondef will not return True indicating that the user has overwritten the default value.
Object	Application.UserOptions.Exists
Syntax	Exists
Parameters	OptionName as String: The name of the option. In combination with OptionCategory, this provides the primary key and must be unique. Maximum length is 64 characters. OptionCategory as String: The Family of the option for categorization. In combination with OptionName, this provides the primary key and must be unique. It does not accept quotation marks as part of the name. Maximum length is 64 characters.
Returns	True if the option exists or false if it does not.
Related Topics	N/A

GetAsBoolean

Exposed In	Version 6.2
Function	Returns the option as a Boolean value.
Object	Application.UserOptions.GetAsBoolean
Syntax	GetAsBoolean

Parameters	OptionName as String: The name of the option. In combination with OptionCategory, this provides the primary key and must be unique. Maximum length is 64 characters. OptionCategory as String: The Family of the option for categorization. In combination with OptionName, this provides the primary key and must be unique. It does not accept quotation marks as part of the name. Maximum length is 64 characters.
Returns	Boolean
Related Topics	N/A

GetAsDateTime

Exposed In	Version 6.2
Function	Returns the option as a Date/Time value.
Object	Application.UserOptions.GetAsDateTime
Syntax	GetAsDateTime
Parameters	OptionName as String: The name of the option. In combination with OptionCategory, this provides the primary key and must be unique. Maximum length is 64 characters. OptionCategory as String: The Family of the option for categorization. In combination with OptionName, this provides the primary key and must be unique. Maximum length is 64 characters.
Returns	Value as Double
Related Topics	N/A

GetAsFloat

Exposed In	Version 6.2
Function	Returns the option as a float value.
Object	Application.UserOptions.GetAsFloat
Syntax	GetAsFloat
Parameters	OptionName as String: The name of the option. In combination with OptionCategory, this provides the primary key and must be unique. Maximum length is 64 characters. OptionCategory as String: The Family of the option for categorization. In combination with OptionName, this provides the primary key and must be unique. It does not accept quotation marks as part of the name. Maximum length is 64 characters.
Returns	Value as Double
Related Topics	N/A

GetAsFont

Exposed In	Version 6.2
Function	Returns the option as a Font value.
Object	Application.UserOptions.GetAsFont
Syntax	GetAsFont

Parameters	OptionName as String: The name of the option. In combination with OptionCategory, this provides the primary key and must be unique. Maximum length is 64 characters. OptionCategory as String: The Family of the option for categorization. In combination with OptionName, this provides the primary key and must be unique. It does not accept quotation marks as part of the name. Maximum length is 64 characters.
Returns	An IFont reference. Refer to Microsoft Developers Network documentation for more information on IFont.
Related Topics	N/A

GetAsInteger

Exposed In	Version 6.2
Function	Returns the option as an Integer.
Object	Application.UserOptions.GetAsInteger
Syntax	GetAsInteger
Parameters	OptionName as String: The name of the option. In combination with OptionCategory, this provides the primary key and must be unique. Maximum length is 64 characters. OptionCategory as String: The Family of the option for categorization. In combination with OptionName, this provides the primary key and must be unique. It does not accept quotation marks as part of the name. Maximum length is 64 characters.
Returns	Long
Related Topics	N/A

GetAsString

Exposed In	Version 6.2
Function	Returns the option as a String value.
Object	Application.UserOptions.GetAsString
Syntax	GetAsString
Parameters	OptionName as String: The name of the option. In combination with OptionCategory, this provides the primary key and must be unique. Maximum length is 64 characters. OptionCategory as String: The Family of the option for categorization. In combination with OptionName, this provides the primary key and must be unique. It does not accept quotation marks as part of the name. Maximum length is 64 characters.
Returns	String
Related Topics	N/A

GetAsStrings

Exposed In	Version 6.2
Function	Returns the option as Strings.
Object	Application.UserOptions.GetAsStrings
Syntax	GetAsStrings

Parameters	OptionName as String: The name of the option. In combination with OptionCategory, this provides the primary key and must be unique. Maximum length is 64 characters. OptionCategory as String: The Family of the option for categorization. In combination with OptionName, this provides the primary key and must be unique. It does not accept quotation marks as part of the name. Maximum length is 64 characters.
Returns	An IStrings reference.

IStrings Collection	Description
Property Count: Integer	Returns the number of strings in the collection. Use the Count property when iterating over all the strings in the list, or when trying to locate the position of a string relative to the last string in the list.
Method Add:	Adds a string at the end of the list. Call Add to add a string to the end of the list. Add returns the index of the new string. `(Function ).Add(item: OleVariant) : Integer`
Method Remove:	This Method deletes a specified string from the list. Index gives the position of the string, where 0 is the first string, 1 is the second string, and so on. `(Procedure). Delete(Index: Integer);`
Method Clear:	This Method empties the list. `(object).Clear;`
Property Item	Returns a string given its index. `(Function) Item(Index: Integer): string;` For example: strings.item[0] returns item 0 in list.

Related Topics	N/A

GetAsVariant

Exposed In	Version 6.2
Function	Returns the option as a Variant value.
Object	Application.UserOptions.GetAsVariant
Syntax	GetAsVariant
Parameters	OptionName as String: The name of the option. In combination with OptionCategory, this provides the primary key and must be unique. Maximum length is 64 characters. OptionCategory as String: The Family of the option for categorization. In combination with OptionName, this provides the primary key and must be unique. It does not accept quotation marks as part of the name. Maximum length is 64 characters.
Returns	Variant
Related Topics	N/A

GetCategory

Exposed In	Version 6.2
Function	Retrieves the options for the current user for the specified category, and adds them to the UserOptions memory cache. This function is used for performance enhancement.
Object	Application.UserOptions.GetCategory
Syntax	GetCategory
Parameters	OptionCategory as String: The Family of the option for categorization. In combination with OptionName, this provides the primary key and must be unique. It does not accept quotation marks as part of the name. Maximum length is 64 characters.
Returns	None
Related Topics	N/A

GetCreateAsBoolean

Exposed In	Version 6.2
Function	Returns the option as a Boolean value. If the default definition value is not created, it creates that entry.
Object	Application.UserOptions.GetCreateAsBoolean
Syntax	GetCreateAsBoolean
Parameters	OptionName as String: The name of the option. In combination with OptionCategory, this provides the primary key and must be unique. Maximum length is 64 characters. OptionCategory as String: The Family of the option for categorization. In combination with OptionName, this provides the primary key and must be unique. It does not accept quotation marks as part of the name. Maximum length is 64 characters. OptionDisplayName as String: "Pretty" name. Maximum length is 64 characters. OptionDefaultValue as Boolean: The value to be stored. In this function, this is the default value that applies to all users unless they have an overriding option of their own.
Returns	Boolean
Related Topics	N/A

GetCreateAsDateTime

Exposed In	Version 6.2
Function	Returns the option as a Date/Time value type. If the default definition value is not created, it creates that entry.
Object	Application.UserOptions.GetCreateAsDateTime
Syntax	GetCreateAsDateTime
Parameters	OptionName as String: The name of the option. In combination with OptionCategory, this provides the primary key and must be unique. Maximum length is 64 characters. OptionCategory as String: The Family of the option for categorization. In combination with OptionName, this provides the primary key and must be unique. It does not accept quotation marks as part of the name. Maximum length is 64 characters. OptionDisplayName as String: "Pretty" name. Maximum length is 64 characters.

	OptionDefaultValue as Double: This is the value to be stored. This is the default value that applies to all users unless they have an overriding option of their own.
Returns	Value as Double
Related Topics	N/A

GetCreateAsFloat

Exposed In	Version 6.2
Function	Returns the option as a Float value type. If the default definition value is not created, it creates that entry.
Object	Application.UserOptions.GetCreateAsFloat
Syntax	GetCreateAsFloat
Parameters	OptionName as String: The name of the option. In combination with OptionCategory, this provides the primary key and must be unique. Maximum length is 64 characters. OptionCategory as String: The Family of the option for categorization. In combination with OptionName, this provides the primary key and must be unique. It does not accept quotation marks as part of the name. Maximum length is 64 characters. OptionDisplayName as String: "Pretty" name. Maximum length is 64 characters. OptionValue as Double: This is the value to be stored. This is the default value that applies to all users unless they have an overriding option of their own.
Returns	Value as Float
Related Topics	N/A

GetCreateAsFont

Exposed In	Version 6.2
Function	Returns the option as a Font. If the default definition value is not created, it creates that entry.
Object	Application.UserOptions.GetCreateAsFont
Syntax	GetCreateAsFont
Parameters	OptionName as String: The name of the option. In combination with OptionCategory, this provides the primary key and must be unique. Maximum length is 64 characters. OptionCategory as String: The Family of the option for categorization. In combination with OptionName, this provides the primary key and must be unique. It does not accept quotation marks as part of the name. Maximum length is 64 characters. OptionDisplayName as String: "Pretty" name. Maximum length is 64 characters. OptionDefaultValue as IFont: This is the value to be stored. This is the default value that applies to all users unless they have an overriding option of their own. Refer to Microsoft Developers Network documentation for more information on IFont.
Returns	Value as IFont.
Related Topics	N/A

GetCreateAsInteger

Exposed In	Version 6.2
Function	Returns the option as an Integer value type. If the default definition value is not created, it creates that entry.
Object	Application.UserOptions.GetCreateAsInteger
Syntax	GetCreateAsInteger
Parameters	OptionName as String: The name of the option. In combination with OptionCategory, this provides the primary key and must be unique. Maximum length is 64 characters.
	OptionCategory as String: The Family of the option for categorization. In combination with OptionName, this provides the primary key and must be unique. It does not accept quotation marks as part of the name. Maximum length is 64 characters.
	OptionDisplayName as String: "Pretty" name. Maximum length is 64 characters.
	OptionDefaultValue as Long: This is the value to be stored. This is the default value that applies to all users unless they have an overriding option of their own.
Returns	Value as Long
Related Topics	N/A

GetCreateAsString

Exposed In	Version 6.2
Function	Returns the option String. If the Default Definition Value does not exist, the entry is created.
Object	Application.UserOptions.GetCreateAsString
Syntax	GetCreateAsString
Parameters	OptionName as String: The name of the option. In combination with OptionCategory, this provides the primary key and must be unique. Maximum length is 64 characters.
	OptionCategory as String: The Family of the option for categorization. In combination with OptionName, this provides the primary key and must be unique. It does not accept quotation marks as part of the name. Maximum length is 64 characters.
	OptionDisplayName as String: "Pretty" name. Maximum length is 64 characters.
	OptionDefaultValue as String: This is the value to be stored. This is the default value that applies to all users unless they have an overriding option of their own.
Returns	Value as String
Related Topics	N/A

GetCreateAsStrings

Exposed In	Version 6.2
Function	Returns the option as multiple Strings. If the default does not exist, the entry is created.
Object	Application.UserOptions.GetCreateAsStrings
Syntax	GetCreateAsStrings
Parameters	OptionName as String: The name of the option. In combination with OptionCategory, this provides the primary key and must be unique. Maximum length is 64 characters.

	OptionCategory as String: The Family of the option for categorization. In combination with OptionName, this provides the primary key and must be unique. It does not accept quotation marks as part of the name. Maximum length is 64 characters. OptionDisplayName as String: "Pretty" name. Maximum length is 64 characters. OptionDefaultValue as iString: This is the value to be stored. This is the default value that applies to all users unless they have an overriding option of their own. See "GetAsStrings" on page 109 for information on the iStrings collection.
Returns	Value as iStrings
Related Topics	N/A

GetCreateAsVariant

Exposed In	Version 6.2
Function	Returns the option as a Variant value. If the default does not exist, the entry is created.
Object	Application.UserOptions.GetCreateAsVariant
Syntax	GetCreateAsVariant
Parameters	OptionName as String: The name of the option. In combination with OptionCategory, this provides the primary key and must be unique. Maximum length is 64 characters. OptionCategory as String: The Family of the option for categorization. In combination with OptionName, this provides the primary key and must be unique. It does not accept quotation marks as part of the name. Maximum length is 64 characters. OptionDisplayName as String: "Pretty" name. Maximum length is 64 characters. OptionDefaultValue as Variant: This is the value to be stored. This is the default value that applies to all users unless they have an overriding option of their own.
Returns	Value as Variant
Related Topics	N/A

GetCreateDefaultAsString

Exposed In	Version 6.2
Function	Gets the actual default value. If the default does not exist, the entry is created.
Object	Application.UserOptions.GetCreateDefaultAsString
Syntax	GetCreateDefaultAsString
Parameters	OptionName as String: The name of the option. In combination with OptionCategory, this provides the primary key and must be unique. Maximum length is 64 characters. OptionCategory as String: The Family of the option for categorization. In combination with OptionName, this provides the primary key and must be unique. It does not accept quotation marks as part of the name. Maximum length is 64 characters. OptionDisplayName as String: "Pretty" name. Maximum length is 64 characters. OptionValue as String: This is the value to be stored. In this function, this is the default value that applies to all users unless they have an overriding option of their own.
Returns	Returns the value as a string value.
Related Topics	N/A

GetDisplayName

Exposed In	Version 6.2
Function	Gets the pretty name.
Object	Application.UserOptions.GetDisplayName
Syntax	GetDisplayName
Parameters	OptionName as String: The name of the option. In combination with OptionCategory, this provides the primary key and must be unique. Maximum length is 64 characters.
	OptionCategory as String: The Family of the option for categorization. In combination with OptionName, this provides the primary key and must be unique. It does not accept quotation marks as part of the name. Maximum length is 64 characters.
Returns	Result as String
Related Topics	N/A

GetLocked

Exposed In	Version 6.2
Function	User cannot change the value unless the connecton string belongs to the Administrator. Determines if the option is locked for the user. If an option is locked, it cannot be written by anyone other than the Administrator.
Object	Application.UserOptions.GetLocked
Syntax	GetLocked
Parameters	OptionName as String: The name of the option. In combination with OptionCategory, this provides the primary key and must be unique. Maximum length is 64 characters.
	OptionCategory as String: The Family of the option for categorization. In combination with OptionName, this provides the primary key and must be unique. It does not accept quotation marks as part of the name. Maximum length is 64 characters.
Returns	Boolean
Related Topics	N/A

Remove

Exposed In	Version 6.2
Function	Removes a definition value for a user option.
Object	Application.UserOptions.Remove
Syntax	Remove
Parameters	OptionName as String: The name of the option. In combination with OptionCategory, this provides the primary key and must be unique. Maximum length is 64 characters.
	OptionCategory as String: The Family of the option for categorization. In combination with OptionName, this provides the primary key and must be unique. It does not accept quotation marks as part of the name. Maximum length is 64 characters.
Returns	None
Related Topics	N/A

SetAsBoolean

Exposed In	Version 6.2
Function	Sets the option as a Boolean value.
Object	Application.UserOptions.SetAsBoolean
Syntax	SetAsBoolean
Parameters	OptionName as String: The name of the option. In combination with OptionCategory, this provides the primary key and must be unique. Maximum length is 64 characters. OptionCategory as String: The Family of the option for categorization. In combination with OptionName, this provides the primary key and must be unique. It does not accept quotation marks as part of the name. Maximum length is 64 characters. OptionValue as Boolean: This is the value to be stored. This is the default value that applies to all users unless they have an overriding option of their own. OptionLocked as Boolean: If the value is locked, only the Administrator can change the value.
Returns	None. Generates an error if the user option does not exist in the default table.
Related Topics	N/A

SetAsDateTime

Exposed In	Version 6.2
Function	Sets the option as a Date / Time value.
Object	Application.UserOptions.SetAsDateTime
Syntax	SetAsDateTime
Parameters	OptionName as String: The name of the option. In combination with OptionCategory, this provides the primary key and must be unique. Maximum length is 64 characters. OptionCategory as String: The Family of the option for categorization. In combination with OptionName, this provides the primary key and must be unique. It does not accept quotation marks as part of the name. Maximum length is 64 characters. OptionValue as Double: This is the value to be stored. This is the default value that applies to all users unless they have an overriding option of their own. OptionLocked as Boolean: If the value is locked, only the Administrator can change the value.
Returns	None
Related Topics	N/A

SetAsFloat

Exposed In	Version 6.2
Function	Sets the option as a Float.
Object	Application.UserOptions.SetAsFloat
Syntax	SetAsFloat
Parameters	OptionName as String: The name of the option. In combination with OptionCategory, this provides the primary key and must be unique. Maximum length is 64 characters. OptionCategory as String: The Family of the option for categorization. In combination with OptionName, this provides the primary key and must be

	unique. It does not accept quotation marks as part of the name. Maximum length is 64 characters. OptionValue as Double: This is the value to be stored. This is the default value that applies to all users unless they have an overriding option of their own. OptionLocked as Boolean: If the value is locked, only the Administrator can change the value.
Returns	None
Related Topics	N/A

SetAsFont

Exposed In	Version 6.2
Function	Sets the option as a Font value.
Object	Application.UserOptions.SetAsFont
Syntax	SetAsFont
Parameters	OptionName as String: The name of the option. In combination with OptionCategory, this provides the primary key and must be unique. Maximum length is 64 characters. OptionCategory as String: The Family of the option for categorization. In combination with OptionName, this provides the primary key and must be unique. It does not accept quotation marks as part of the name. Maximum length is 64 characters. OptionValue as IFont: This is the value to be stored. This is the default value that applies to all users unless they have an overriding option of their own. IFont is a standard Microsoft interface. For information refer to the Microsoft Developers Network documentation. OptionLocked as Boolean: If the value is locked, only the Administrator can change the value.
Returns	None
Related Topics	N/A

SetAsInteger

Exposed In	Version 6.2
Function	Sets the option as an Integer.
Object	Application.UserOptions.SetAsInteger
Syntax	SetAsInteger
Parameters	OptionName as String: The name of the option. In combination with OptionCategory, this provides the primary key and must be unique. Maximum length is 64 characters. OptionCategory as String: The Family of the option for categorization. In combination with OptionName, this provides the primary key and must be unique. It does not accept quotation marks as part of the name. Maximum length is 64 characters. OptionValue as Long: This is the value to be stored. This is the default value that applies to all users unless they have an overriding option of their own. OptionLocked as Boolean: If the value is locked, only the Administrator can change the value.
Returns	None
Related Topics	N/A

SetAsString

Exposed In	Version 6.2
Function	Sets the option as a String value.
Object	Application.UserOptions.SetAsString
Syntax	SetAsString
Parameters	OptionName as String: The name of the option. In combination with OptionCategory, this provides the primary key and must be unique. Maximum length is 64 characters. OptionCategory as String: The Family of the option for categorization. In combination with OptionName, this provides the primary key and must be unique. It does not accept quotation marks as part of the name. Maximum length is 64 characters. OptionValue as String: This is the value to be stored. This is the default value that applies to all users unless they have an overriding option of their own. OptionLocked as Boolean: If the value is locked, only the Administrator can change the value.
Returns	None
Related Topics	N/A

SetAsStrings

Exposed In	Version 6.2
Function	Sets the option as Strings value.
Object	Application.UserOptions.SetAsStrings
Syntax	SetAsStrings
Parameters	OptionName as String: The name of the option. In combination with OptionCategory, this provides the primary key and must be unique. Maximum length is 64 characters. OptionCategory as String: The Family of the option for categorization. In combination with OptionName, this provides the primary key and must be unique. It does not accept quotation marks as part of the name. Maximum length is 64 characters. OptionValue as String: This is the value to be stored. This is the default value that applies to all users unless they have an overriding option of their own. OptionLocked as Boolean: If the value is locked, only the Administrator can change the value.
Returns	None
Related Topics	N/A

SetAsVariant

Exposed In	Version 6.2
Function	Sets the option as a Variant value.
Object	Application.UserOptions.SetAsVariant
Syntax	SetAsVariant

Parameters	OptionName as String: The name of the option. In combination with OptionCategory, this provides the primary key and must be unique. Maximum length is 64 characters. OptionCategory as String: The Family of the option for categorization. In combination with OptionName, this provides the primary key and must be unique. It does not accept quotation marks as part of the name. Maximum length is 64 characters. OptionValue as Variant: This is the value to be stored. This is the default value that applies to all users unless they have an overriding option of their own. OptionLocked as Boolean: If the value is locked, only the Administrator can change the value.
Returns	None
Related Topics	N/A

SetDefaultAsString

Exposed In	Version 6.2
Function	Sets the default value as a String.
Object	Application.UserOptions.SetDefaultAsString
Syntax	SetDefaultAsString
Parameters	OptionName as String: The name of the option. In combination with OptionCategory, this provides the primary key and must be unique. Maximum length is 64 characters. OptionCategory as String: The Family of the option for categorization. In combination with OptionName, this provides the primary key and must be unique. It does not accept quotation marks as part of the name. Maximum length is 64 characters. OptionValue as String: This is the value to be stored. This is the default value that applies to all users unless they have an overriding option of their own.
Returns	None
Related Topics	N/A

SetLocked

Exposed In	Version 6.2
Function	Locks the option from writing.
Object	Application.UserOptions.SetLocked
Syntax	SetLocked
Parameters	OptionName as String: The name of the option. In combination with OptionCategory, this provides the primary key and must be unique. Maximum length is 64 characters. OptionCategory as String: The Family of the option for categorization. In combination with OptionName, this provides the primary key and must be unique. It does not accept quotation marks as part of the name. Maximum length is 64 characters. OptionValue as Long: This is the value to be stored. This is the default value that applies to all users unless they have an overriding option of their own. OptionLocked as Variant: If the value is locked, only the Administrator can change the value.
Returns	None
Related Topics	N/A

UserID

Exposed In	Version 6.2
Function	Gets/sets the UserID that this option object should operate. By default, this function is set to the current logged in user.
Object	Application.UserOptions.UserID
Syntax	UserID
Parameters	UserID as string
Returns	The ID for the current logged in user.
Related Topics	N/A

Application.Users

This object contains a collection of Sage SalesLogix users.

Count

Function	Read-only. Returns the number of Sage SalesLogix users.
Object	Application.Users.Count
Syntax	Count
Parameters	Value as Long
Returns	Long
Related Topics	N/A

Item

Function	Read/write. Returns a single User object.
Object	Application.Users.Item Also see "User Object" on page 141.
Syntax	Item()
Parameters	Index to a User object as Integer (0..Count-1), or ID of a user as String.
Returns	User object
Related Topics	N/A

Application.Version

Function	This property returns a string representation of the Sage SalesLogix version, for example, "6.0.0.3185" (original release of Sage SalesLogix v6.0).
Object	Application.Version
Syntax	Version
Parameters	None
Returns	String
Related Topics	N/A

Chapter 2
Scripting Properties and Functions

The following properties and functions are available through scripting. The functions and properties apply only to the object under which they are listed.

Application Object Events

ActivityListWindowClose(ActivityListWindow)

Invoked after an ActivityListWindow object is closed.

ActivityListWindowOpen(ActivityListWindow)

Invoked after an ActivityListWindow object is generated.

AfterCompleteActivity(Activity, HistoryID)

This function takes an Activity object and a new HistoryID.

AfterCreateHistory(Activity, Sender)

Invoked after a History item is generated. Sender will contain a reference to the object that called the function.

AfterDeleteActivity(ActivityID)

Invoked after an Activity is deleted. Note that an Activity object is not passed because the Activity no longer exists.

AfterEditActivity(Activity)

Invoked after an Activity is edited.

AfterPostActivity(Activity, RecordSet)

Invoked after an Activity (insert or update) gets posted to the database. The Recordset object contains a recordset pointing at the database Activity record.

BeforeCompleteActivity(Activity, Cancel)

This is invoked just before an Activity is completed. Setting Cancel to True will prevent the completion taking place.

BeforeCreateHistory(Activity, Sender)

Invoked just before the History is created. Sender is an object reference to the caller.

BeforeDeleteActivity(Activity, Cancel)

Invoked just before an Activity is deleted. Setting Cancel = True will stop the deletion.

BeforeEditActivity(Activity, Cancel)

Invoked just before an Activity is edited. Setting Cancel = True will stop the edit.

BeforePostActivity(Activity, RecordSet)

Invoked just before the Activity gets posted to the database. The Recordset contains the data that is about to be posted - this can be modified prior to posting.

LogonComplete()

This is invoked after the user has completed a logon.

Quit()

This is invoked when the application closes.

Startup()

This is invoked on startup - just at the point where the application first comes to the screen, but before the point when the login process is complete.

ReceiveOutlookMessage(Message, EmailLogging)

This gets invoked when an e-mail message is received. The Message object contains a MailMessage.

SendOutlookMessage(Message, EmailLogging)

This gets invoked when an e-mail message is sent. The Message object contains a MailMessage.

MailMessage Class

The MailMessage class is used by the SendOutlookMessage and ReceiveOutlookMessage functions.

Properties:

Attachments	(MailAttachments) List of attachments.
BCC	(String) BCC address line.
Body	(String) Body of the message.
CC	(String) CC address line.
FlagRequest	(String) Flag.
ReceivedByName	(String) Name of received-by.
ReceivedTime	(String) Time received.
Recipients	(MailRecipients) List of recipients.
RelatedRecords	(RelatedRecords) The list of related records.
SenderAddressType	(String) The type of address.
SenderEMailAddress	(String) The sender’s e-mail address.
SenderName	(String) Name of the sender.
Subject	(String) Subject of the message.
To	(String) To address line.

MailRecipients Class

The MailRecipients class is a collection of MailRecipient classes.

Property:

Count	(Long Integer) The number of MailRecipient objects in the list.

Method:

Item	(Index Long) A MailRecipient class.

MailRecipient Class

This contains details of the mail recipients.

Properties:

Address	(String) The address of the recipient.
AddressType	(String) The type of address.
Name	(String) The name of the recipient.
Type	(TxMailRecipientType) The type of recipient.

TxMailRecipientType

rtTo	1
rtCC	2
rtBCC	3

MailAttachments Class

The MailAttachments class is a collection of MailAttachment classes.

Properties:

Count	(Integer Long) The number of MailAttachment objects in the list.

Method:

Item	(Index Long) A MailAttachment class.

MailAttachment Class

This class represents one e-mail attachment.

Properties:

FileName	(String) The name of the file.
Size	(Long) Size of the file.

Methods:

SaveAsFile	(FileName as String) Saves the attachment.

SaveToSlx	(ContactID as String, AccountID as String, OpportunityID as String, TicketID as String, HistoryID as String, [LeadID as String], [OtherEntityTableName as String], [OtherEntityKeyValue as String]) Saves the attachment to the attachments table and returns the AttachID.

RelatedRecords Class

This is a class to hold a list of all the records that are related to an e-mail message.

Property:

Count	(Long) Number of Related Records.

Methods:

Add	(TableName as String, ID as String) Adds a new RelatedRecord class.
Delete	(Index as Long) Deletes the index provided.
Item	(Index as Long) Returns the RelatedRecord at the index.

RelatedRecord Class

The RelatedRecord class holds the details of the Related Records.

Properties:

ID	(String) The ID of the related record.
TableName	(String) The table name.

Activity Object

Represents one Activity.

Properties:

AccountID	(String) The ID of the account associated with the activity.
AccountName	(String) The display name of the account associated with the activity.
Alarm	(Boolean) Indicates the Alarm is on when True.
AlarmTime	(DateTime) Time the alarm is set to trigger.
Attachments	(Attachments Read Only) The collection of attachments. When setting this property, all attachments from the target activity are replaced by the attachments from the source activity by duplicating the corresponding rows from the ATT ACHMENT table.
Attendees	(Read Only:AttendeeList) List of attendees.
BasedOn	(Read Only: String) Previous activity to which the current activity is linked.
Category	(String) The activity category.
Completed	(Read Only: Boolean) Indicates the activity is complete when True.
CompleteDate	(DateTime) Date of completion.
Confirmed	(Read Only: Boolean) Indicates the activity is confirmed when True.
ContactID	(String) The ID of the contact to which the activity is associated.
ContactName	(String) The display name of the contact.

CreateDate	(DateTime) Date the activity was created.
Declined	(Read Only: Boolean) Indicates the activity is declined when True.
Duration	(Integer - long) Number of minutes for which the activity is scheduled.
EndTime	(Read Only: DateTime) Time the activity is scheduled to end.
ForCurrentUser	(Read Only: Boolean) Indicates the activity is for the current user when True.
ForeignID	(String) Used to associate the activity with other tables.
Key	(Read Only: String) The ID of the activity.
LongNotes	(String) Notes associated with the activity.
Modified	(Boolean) Indicates the activity is modified when True.
ModifyDate	(DateTime) Date the activity was last modified.
Notes	(String) Notes for the activity shortened to 255 characters for display.
OpportunityID	(String) The ID of the opportunity associated with the activity.
OpportunityName	(String) The name displayed for the opportunity.
OriginalDate	(DateTime) First date for which the activity is scheduled.
Personal	(Boolean) Indicated a personal activity when True.
Priority	(String) The priority of the activity.
Recurring	(Boolean) Indicates a recurring activity when True.
Resources	(Read Only: ResourceList) List of resources associated with the activity.
ResultCode	(String) The result code of the activity.
ResultString	(String) The result string of the activity.
StartTime	(DateTime) The time the activity is scheduled to begin.
TicketID	(String) The ID of the ticket associated with the activity.
Timeless	(Boolean) Indicates the activity is timeless when True.
Title	(String) The activity description.
Type	(Read Only: TxActivityType) The activity type.
User	(Read Only: User) The user associated with the activity.
UserID	(String) The ID of the user associated with the activity.

Methods:

Complete	(ShowDialog: Boolean) Completes the activity.
Delete	Deletes the activity.
Display	(Read Only: Boolean) Displays the activity.
Refresh	Refreshes the activity details.
Save	Saves the activity.

ActivityList Class

This is a list of activities returned by the GetActivityList function.

Property:

Count	(Long) The number of activities in the list.

Method:

Item	(Index as Long) An Activity class.

txActivityType

atMeeting	0
atPhoneCall	1
atToDo	2
atPersonal	3
atInternal	4
atEmail	5
atFax	6
atProcess	7
atLiterature	8
atNote	9

AttendeeList Class

This class holds details of the users that are linked to an activity.

Properties:

Count	(Read Only, Integer) The number of attendees.

Functions:

Add	(UserID as String) Allows you to add a new user as an attendee.
Item	(Index as Long) Retrieves the ActivityAttendee class for the selected item.

ActivityAttendee Class

This class maintains information about each Attendee to an activity.

Properties:

Confirmed	(String) Confirmed attendee - T/F.
User	(Read Only, String) Name of the user.
UserID	(String) User ID of the user.

Resource Class

This class represents an individual resource in the resource list.

Properties:

Name	(String) The name of the resource.
ResourceID	(String) The ID of the resource.
Type	(String) The type of resource.

ResourceList Class

This class allows us to manage the resources that are associated with an activity.

Property:

Count	(Long) The number of resources in the list.

Methods:

Add	(ResourceID as String) Add a new resource to the list.
Item	(Index as Long) Return the Resource class at the passed index.

Active Form

Properties:

Parent	(Object) indicates the Parent Main View Object.
PluginName	Provides the plugin name of the form [Form.PluginName could return 'Personal:C_frmAdds']
PluginID	Provides the 12 character plugin ID of the form. [Form.PluginID could return 'pQF8AA0007DW']
Script	Returns an object representing the script behind the form with the functions and subroutines exposed as that object's methods, and with global script variables exposed as properties. [Form.Script.C_CheckDates would execute subroutine 'C_CheckDates' defined under the form Script tab]
IsReading	Boolean. Read-only. This property is set to True when the form controls are being populated with the database data, rather than by a user action. The property is set to False when the form controls are being populated by a user action.
IsWriting	Boolean. Read-only. This property is set to True before the data is posted to the database. It is reset to False immediately afterwards. Since currently there are no events firing in that time interval, you will not see this in the script. If you create a custom control, you might see IsWriting set to True when a bound property is being read.
IsValidating	Boolean. Read-only. This property is set to True when form data is being validated. It is set to False when form data is not being validated.
Modified	Boolean. Read-only. Accessing this property causes the form to loop through all of its bound controls and compare the current values of the bound properties with the corresponding values that were retrieved from the database when the controls were populated with data. If a mismatch is found, Modified returns True.

ActivityDialog Object

Permits you to control the Activities Detail dialog box:

Properties:

Activity	(Activity, Read Only) Returns the current Activity object.
Caption	(String) The dialog caption.
Height	(Integer) The height of the Activity Dialog box.
Left	(Integer) Specifies the left edge of the Activity Dialog box.
TabAttachments	(Read Only) Returns the ActivityTab object representing the "Attachments" tab. (Also see the Tabs collection.)
TabGeneral	(Read Only) Returns the ActivityTab object representing the "General" tab. (Also see the Tabs collection.)
TabMembers	(Read Only) Returns the ActivityTab object representing the "Members" tab. (Also see the Tabs collection.)
TabOutlookFreeBusy	(Read Only) Returns the ActivityTab object representing the"OutlookFreeBusy" tab. (Also see the Tabs collection.)
TabRecurrence	(Read Only) Returns the ActivityTab object representing the"Recurrence" tab. (Also see the Tabs collection.)
TabResources	(Read Only) Returns the ActivityTab object representing the "Resources" tab. (Also see the Tabs collection.)
Tabs	(Read Only) Returns a collection containing all activity dialog tabs.
Top	(Integer) Specifies the top of the Activity Dialog box.
Width	(Integer) Specifies the width of the Activity Dialog box.

Functions:

Cancel	Same as clicking the Cancel button.
CloseAndSave	Same as clicking the OK button. Returns True if the dialog was closed and the activity was successfully saved. Otherwise the dialog stays open and the activity is not saved.

ActivityListFilter Object

Represents a particular filter.

Properties:

Name	(String, Read Only) Returns the name of the filter.
Start	(Date, Read Only) The starting date of the filter.
End	(Date, Read Only) The ending date of the filter.

Methods:

Delete	Removes the filter.
Apply	Selects the filter in the combo box and applies it.
Related Topic:	ActivityListFilters Object

ActivityListFilters Object

Properties:

Count	(Integer, Read Only) Returns the number of filters in the collection.

Functions:

Add	(FileName, DisplayName, ShowDialog) Adds a new attachment with the specified file name. Filename - Required. The fully qualified path of the attachment to add. DisplayName - Optional. If specified, the attachment display name, othewise the display name will be the file name (minus path) of the attachment. ShowDialog - Optional. If True, the Add attachment dialog box appears. Returns Attachment object.
AddAll	Adds and returns an ActivityListFilter object corresponding to the "All" filter.
AddToday	Adds and returns an ActivityListFilter object corresponding to the "Today" filter.
AddTomorrow	Adds and returns an ActivityListFilter object corresponding to the "Tomorrow" filter.
AddNextMonth	Adds and returns an ActivityListFilter object corresponding to the "Next Month"filter.
AddNextQuarter	Adds and returns an ActivityListFilter object corresponding to the "Next Quarter" filter.
AddNextWeek	Adds and returns an ActivityListFilter object corresponding to the "Next Week" filter.
AddThisMonth	Adds and returns an ActivityListFilter object corresponding to the "This Month" filter.
AddThisQuarter	Adds and returns an ActivityListFilter object corresponding to the "This Quarter" filter.
AddThisWeek	Adds and returns an ActivityListFilter object corresponding to "This Week" filter.
AddPast	Adds and returns an ActivityListFilter object corresponding to the "Past" filter.
AddDateRange(Name, Start, End)	Adds and returns an ActivityListFilter object. Name - (String) Name of the ActivityListFilter object. Start - (DateTime) The start of the time range of the filter. End - (DateTime) The end of the time range of the filter.
AddDateRangePrompt	Adds and returns an ActivityListFilter object corresponding to the "Date Range..." filter. The user is prompted to enter the range.
Clear	Removes all filters.

ActivityListTab Object

Properties:

Active	(Boolean) Determines whether the given tab is the active tab. Setting the property to True activates the tab.
Caption	(String) Returns or sets the caption of the tab.
Kind	(TxActivityListTabKind, Read Only) The kind of tab. enum: altkAll (0), altkCalls (1), altkMeetings (2), altkTodos (3), altkPersonal (4), altkAlarms (5), altkEvents (6), altkLiterature (7), altkConfirmations (8).
Visible	(Boolean, Read/Write) Determines whether the tab is visible in the Activity List Main view.
Related Topic:	ActivityListWindow object

ActivityListTabs Object

Properties:

ActiveTab	(ActivityListTab, Read Only.) The active tab.
Count	(Integer, Read Only.) Returns the number of tabs.
TabAll	(ActivityListTab, Read Only.) The "All Open" tab.
TabCalls	(ActivityListTab, Read Only.) The "Calls" tab.
TabMeetings	(ActivityListTab, Read Only.) The "Meetings" tab.
TabTodos	(ActivityListTab, Read Only.) The "Todos" tab.
TabPersonal	(ActivityListTab, Read Only.) The "Personal" tab.
TabAlarms	(ActivityListTab, Read Only.) The "Alarms" tab.
TabEvents	(ActivityListTab, Read Only.) The "Events" tab.
TabLiterature	(ActivityListTab, Read Only.) The "Literature" tab.
TabConfirmations	(ActivityListTab, Read Only.) The "Confirmations" tab.

Functions:

Item(Index)	Returns an ActivityListTab object with the given index (0 through Count-1).
Related Topic:	ActivityListWindow object

ActivityListWindow Object

Properties:

ActiveTab	(ActivityListTab) The currently active tab.
Activities	(ActivityList) The ActivityList collection of activities selected in the active tab.
CalendarUsers	(CalendarUsers) The collection of users whose calendars are displayed by the Network Client. Refer to "CalendarUsers Object" on page 135.
Filters	(ActivityListFilters) The collection of all activities in the active tab.
PreviewPaneForm	(Object) The object representing the AX preview form (Activity:Activity Preview); this is the same object accessible through the Form intrinsic variable in the Form's VB script.

PreviewPaneHeight (Integer, Read/Write.) The height of the preview pane in pixels.

SelectedActivities (ActivityList) The collection of all activities selected in the active tab.

Tabs (ActivityListTabs) The collection of tabs, both visible and invisible.

Method:

Close Closes the Activities List Main view.

Examples:

This example shows hiding tabs, changing the tab captions, and modifying the filters.

```
'Hide two tabs and change the captions of another one
'Can use the parameter passed to the event handler property or
'use the Application.ActivtyListWindow property
        Window.Tabs.TabLiterature.Visible = false
        Application.ActivityLIstWindow.Tabs.TabToDos.Visible = false
        Window.Tabs.TabConfirmations.Caption="Invitations && Notifications"
'Rebuild the time range filters from scratch
'The list does not need to be cleared, but will do anyway
        Application.ActivityLIstWindow.Filters.Clear
        Application.ActivityLIstWindow.Filters.AddAll
        Application.ActivityLIstWindow.Filters.AddToday
        Application.ActivityLIstWindow.Filters.AddTomorrow
        Application.ActivityLIstWindow.Filters.AddThisWeek
        Application.ActivityLIstWindow.Filters.AddDateRange "Next 7 days", DateAdd ("d",
         1, Date), DateAdd ("d", -7, Date)
```

Enumerate all selected activities.

```
dim Act
for each Act in Application.ActivityListWindow.SelectedActivities
MsgBox Act.Title
Next
```

Enumerate all activities displayed in the active tab.

```
dim Act
for each Act in Application.ActivityListWindow.Activities
MsgBox Act.Title
Next
```

Display a particular tab.

```
Application.ActivityListWindow.Tabs.TabConfirmations.Active = true
```

Add a user to the list of users whose calendar is displayed.

```
Application.ActivityListWindow.CalendarUsers.Add ( Application.Users(“lee”) )
```

Expand the preview pane height.

```
Application.ActivityListWindow.PreviewPaneHeight =
Application.ActivityListWindow.PreviewPaneHeight + 20
```

Access a control in the AX preview pane.

```
MsgBox Application.ActivityListWindow.PreviewPaneForm.MemNotes.Text
```

Apply a date filter.

```
dim Filter
set Filter = Application.ActivityListWindow.Filters (“Next week”)
Filter.Apply
```

ActivityTab Object

Properties:

Active	(Boolean) Determines whether the given tab is the active tab.
Caption	(String) The tab caption.
Kind	(Read Only) TxActivityTabKind enum: atkDetails (0), atkMembers (1), atkRecurrence (3), atkOutlookFreeBusy (4).
Visible	(Boolean) Determines whether the tab is visible.

ActivityTabs Object

Properties:

ActiveTab	Returns or sets the active tab (ActivityTab object).
Count	Returns the number of tabs.

Functions:

Item(Index)	Returns a tab (ActivityTab object) with the given index (0 through Count-1).

Attachment Object

Properties:

AttachID	String, Read Only.
AttachDate	Date, Read/Write.
AccountID	String, Read/Write.
ContactID	String, Read/Write.
OpportunityID	String, Read/Write.
Description	String, Read/Write.
FileSize	Integer, Read Only.

FileName	String, Read/Write.
UserID	String, Read/Write
ContractID	String, Read/Write.
DocumentType	String, Read/Write.
ProcedureID	String, Read/Write.
ProductID	String, Read/Write.
RMAID	String, Read/Write.
TicketID	String, Read/Write.
HistoryID	String, Read/Write.
ModifyDate	Date, Read Only.
ModifyUser	String, Read Only.
CreateDate	Date, Read Only.
CreateUser	String, Read Only.
LeadID	String, Read Only.
ActivityID	String, Read Only.

Functions:

Delete	Deletes the attachment.
Save	Saves the newly added or modified attachment.
Related Topic:	Attachments object

Attachments Object

Properties:

Count	(Integer, Read Only.) Returns the number of attachments in the collection.

Functions:

Add(FileName, DisplayName, ShowDialog)	Adds a new attachment with the specified file name. Filename - Required. The fully qualified path of the attachment to add. DisplayName - Optional. If specified, the attachment display name, otherwise the display name will be the file name (minus path) of the attachment. ShowDialog - Optional. If True, the Add attachment dialog box appears. Returns an Attachment object.
Item(Index)	Returns a tab (ActivityTab object) with the given index (0 through Count-1).
Remove	Removes an attachment with the given index (0 through Count-1).
Related Topic:	Attachment object

CalendarUsers Object

Derived from the Users object and inherits Item(Index) and Count properties.

Methods:

Add(User, Color)	Adds the given User object to the calendar users. User - (User) The user to add. Color - (Integer) Optional. Specifies the RGB value of the user's activities.
Remove(Index)	Removes the calendar user with the given index from the list of calendar users. Index - (Integer) Index 0 through Count.
Colors(Index)	Returns the RGB value corresponding to the user with the given index, (0 through Count).
Related Topic:	User object

DataGridColumns Object

Properties:

Item(Index)	Returns the number of columns in the DataGrid.
Count	Provides access to the list of DataGrid columns.

Functions:

Add(Type)	Adds and returns a new column.

DataGrid Object

Properties:

AllowNulBindID Controls whether the DataGrid issues a query when BindID is empty. DataGrids are normally bound through the BindID property and issue a query (load the data) any time the BindID property changes. DataGrids may also issue a query when the page loads, depending on the values of BindID and AllowNullBindId. Applies only to changes to the BindId property. Explicitly setting the SQL property or calling the Refresh method will re-query even if BindId is empty and AllowNullBindId is set to False.

Value	Description
True	When the page loads, the DataGrid issues a query, even if BindID is empty.
False	When the page loads, the DataGrid does not issue a query if BindID is empty. This eliminates an unnecessary load of all records when the page loads before BindId is set.

Columns	Returns the grids columns collection.
PopupMenu	Identifies the popup menu associated with a data grid.
SQL	Returns the SQL object.

Functions:

GetCurrentField Returns the value of the specified field of the currently selected row. If FieldName is an empty string, KeyField (if specified) is assumed.

GetFieldValue Returns the value of a specified field of a data grid.

Parameters	Description
KeyFieldValue	The ID of the of the KeyField property in a data grid.
ColumnName	The name of the column from which data will be returned.

GetHitInfo(X,Y) Used in the MouseUp event to determine which node in the datagrid was clicked.

Example:

```
Option Explicit
Sub DataGrid1MouseUp(Sender, Button, X, Y)
  if Button = 0 Then 'mbLeft
    dim HitInfo
    dim HitType
    dim Column
    dim Node
    set HitInfo = DataGrid1.GetHitInfo(X, Y)
    set Column = HitInfo.Column
    set Node = HitInfo.Node
    if (Not (Column Is Nothing)) and (Not (Node Is Nothing)) and
       (HitInfo.HitType = 7) Then 'htLabel = 7 - click on a node
      If (Node.ValuesCount > 0) Then
        Memo1.Lines.Add "FieldName: " & Column.FieldName & " Value: " &
         Node.Values(Column.ColIndex)
      End If
    End If
  End If
End Sub
```

Refresh Refreshes the grid.

SetFieldValue Sets the value of a specified field of a datagrid during runtime.

Parameter	Description
KeyFieldValue	The ID of the of the KeyField property in a data grid.
ColumnName	The name of the column from which data will be returned.
Value	The new value you are setting.
UpdateRecordset	(Optional) Determines whether the change should be kept in sync with the dataset. The default value is True.

ListColumn Object

Properties:

Alignment Specifies how all text is aligned within the list column.

Autosize Specifies whether the list column automatically sizes itself to the width of its text.

Caption Specifies the text that appears at the top of the column.

ImageIndex The index of the image displayed in the column.

MaxWidth	Specifies the maximum column width.
MinWidth	Specifies the minimum column width.
Width	Specifies the width of the column.
WidthType	(Read Only) Indicates whether the column is sized automatically. The read-only WidthType property indicates how the column width is determined. WidthType is set to the same value as Width. WidthType, however, retains its negative value when Width changes automatically. If WidthType returns -1, the list column is automatically resized to accommodate the text in the column. If WidthType returns -2, the list column is automatically resized to accommodate the column header. If WidthType returns a nonnegative value, the column is not resized automatically; in this case, the value of Width and WidthType should be the same. To enable automatic column resizing, assign the value -1 or -2 directly to Width.

ListColumns Object

Properties:

Count	Returns the number of columns in the ListView.
Item(Index)	Lists the columns in the collection. Returns ListColumn object.

Functions:

Add	Creates and returns a new ListColumn instance and adds it to the Items array.
Delete(Index)	Deletes a single ListColumn from the ListView.

ListItem Object

Properties:

Checked	Determines whether a check mark appears next to the list item.
Cut	Determines if the list item is drawn as if it is selected for a cut operation.
Data	Specifies any application-specific data associated with the list item.
Focused	Indicates whether the list item has input focus.
Handle	Specifies the window handle of the list view that owns the list item.
ImageIndex	Determines which image is displayed as the icon for the list item.
Indent	Specifies how far the list item is indented.
Index	Indicates the position of the list item in the ListItems collection.
OverlayIndex	Determines which image from the image list is used as an overlay mask.
Left	Specifies the distance, in pixels, from the left edge of the list view to the left edge of the list item.
Selected	Indicates whether the list item is selected.
StateIndex	Specifies which image from the StateImages image list to display for the item.
SubItems	Contains any strings that appear as subitems to the list item.
Top	Specifies the distance, in pixels, from the top of the list view to the top of the list item.
Tag	Specifies any application-specific data associated with the list item.

SubItemImages	Indicates which images (if any) should appear next to subitems of the item.

Functions:

CancelEdit	Cancels the editing of the list item's caption.
Delete	Deletes the list item from its list view.
EditCaption	Begins in-place editing of the list item's caption.
MakeVisible	Scrolls the list view, if necessary, to ensure a list item is in view.
WorkArea	Returns the work area (if any) that contains the list item.
Update	Updates the image of the list item in the list view display.

ListItems Object

Properties:

Count	Indicates the number of items in the Item property array.
Handle	Specifies the window handle for the list view that displays the items managed by ListItems.
Items(Index)	Lists all list items managed by the ListItems object.

Functions:

Add	Creates and returns a new list item and adds it to the ListView control.
Clear	Removes all items from the list.
Delete(Index)	Deletes a specified item and updates the list view accordingly.
Insert(Index)	Creates and returns a new list item and inserts it into the list view.

Node Object

Properties:

AbsoluteIndex	Indicates the index of the tree node relative to the first tree node in a tree node.
Count	Indicates the number of direct descendants of a tree node.
Cut	Indicates if the tree node object is drawn as if selected as part of a cut and paste operation.
Data	Points to application-defined data associated with the tree node.
Deleting	Indicates whether a node is in the process of being deleted.
Expanded	Specifies whether the tree node is expanded.
Focused	Indicates whether the node appears to have focus.
Handle	Contains the window Handle of the tree view that contains the node.
HasChildren	Indicates whether a node has any children.
ImageIndex	Specifies which image is displayed when a node is in its normal state and is not currently selected.
Index	Specifies the position of the node in the list of child nodes maintained by its parent node.
ItemID	Contains a handle of type HTreeItem that uniquely identifies each node in a tree view.

Level	Indicates the level of indentation of a node within the tree view control.
OverlayIndex	Determines which image from the image list is used as an overlay mask.
Parent	Identifies the parent node of the tree node.
Selected	Determines whether the node is selected.
Tag	Points to application-defined data associated with the tree node.

Nodes Object

This object represents a collection of nodes.

Properties:

Count	Indicates the number of nodes maintained by the TreeView object.
Item(Index)	Lists all tree nodes managed by the TreeView object.

Functions:

AlphaSort	Sorts the nodes children alphabetically based on their Text property.
Collapse	(Recurse) Collapses a node.
Delete	Destroys the node and all its children.
DeleteChildren	Deletes all children of the node.
EditText	Begins in-place editing of the specified node's text, replacing the text of the node with a single-line edit control containing the text.
EndEdit	Ends the editing of a node's label.
Expand	(Recurse) Expands the node to display all child nodes.
GetFirstChild	Returns the first child node of a tree node.
GetLastChild	Returns the last immediate child node of the calling node.
GetNext	Returns the next node after the calling node in the tree view.
GetNextChild	(Value) Returns the next child node after Value.
GetNextSibling	Returns the next node in the tree view at the same level as the calling node.
GetNextVisible	Returns the next visible node in the tree view after the calling node.
HasAsParent	(Value) Returns True if Value is a parent node of the calling node.
IndexOf	(Value) Returns the position of an immediate child node of the calling node.
MakeVisible	Expands the parent nodes of a node.
MoveTo	(Destination, Mode) Moves the node to another location in the tree view. See TxNodeAttachMode.

Functions:

Add	(Node, Text) Adds a new tree node to a TreeView control. The node is added as the last sibling of the Node parameter.
AddChild	(Node, Text) Adds a new tree node to a tree view. The node is added as a child of the node specified by the Node parameter.
AddChildFirst	(Node, Text) Adds a new tree node to a tree view. Use AddChildFirst to insert a node as the first child of the node specified by the Node parameter.
AddFirst	(Node, Text) Adds a new tree node to a tree view. The node is added as the first sibling of the node specified by the Node parameter.
BeginUpdate	Prevents the updating of the tree view until the EndUpdate method is called.

EndUpdate	Re-enables screen repainting and node reindexing that was turned off with the BeginUpdate method.
Clear	Deletes all tree nodes contained from the list managed by the TreeView.
Delete	(Node) Removes a node from the tree view.
GetFirstNode	Returns the first tree node in the tree view.
Insert	(Node, Text) Inserts a tree node into the tree view before the node specified by the Node parameter.

PopupMenu Object

Properties:

Handle	(Read Only) Provides access to the Windows menu handle for the menu (HMENU).

Functions:

Popup(X, Y)	Displays the pop-up menu on screen. Both parameters are optional. If used, the popup will be shown at the specified screen coordinates, otherwise at the current mouse cursor position.

Reports Collection (IReport)

Properties:

PluginID	PluginID of the report.
UserField	Master User Field as defined in the Report Properties dialog in the Architect (used for filtering).
DateField	Master Date Field as defined in the Report Properties dialog in the Architect (used for filtering).
UseGroupFilter	Show Group Filter as defined in the Report Properties dialog in the Architect.
UseDateFilter	Show Date Filter as defined in the Report Properties dialog in the Architect.
UseUserFilter	Show User Filter as defined in the Report Properties dialog in the Architect.
NativeObject	Reference to Crystal RDC object - same as GetCrystalReport.
MainTable	Main Table as defined in the Report Properties dialog in the Architect (used for filtering).

Functions:

GetRecordSelectionForMainViewCurrentGroup(cont TableName: WideString)

Gets Crystal Syntax formatted RecordSelectionFormula using the conditions of the current main view group for the tables specified.

GetRecordSelectionFormulaForGroup(GroupName or ID: String)

Gets Crystal Syntax formatted RecordSelectionFormula for any SLX Group.

Example:

(More sample code in System:SLX Crystal Report VBScript)

```
    'get instance of Account Summary IReport
```

```
Dim objReport
Set objReport = Application.Reports.Item("Account:Account Summary - Sample")
```

SQL Object

Properties:

Text	SQL query string

Tab Object

Properties:

Highlighted	Indicates whether the tab sheet appears highlighted.
PageIndex	Indicates the index of the tab sheet in the list of tab sheets maintained by the tab control.
TabIndex	(Read Only) Indicates the position of the tab sheet in the set of visible tabs in a TabControl object.
TabVisible	Specifies whether the tab of the Tab object appears in its TabControl.
Caption	Specifies tabs caption.

TabControl Object

Properties:

ActiveTab	Specifies the tab currently displayed by the tab control.
Tabs	(Index) Lists all the tabs in the TabControl object.
TabCount	Indicates the number of tabs in the TabControl object.

Functions:

FindNextPage	(CurPage, GoForward, CheckTabVisible) Returns the next visible tab before or after a specified tab.
SelectNextPage	(Go Forward) Changes the ActiveTab to the first visible tab that is before or after the currently active tab.
ShowControl	(Control) Displays a tab on which the specified control resides.
RowCount	Returns the number of rows in a multi-line tab control.
ScrollTabs	(Delta) Scrolls the tabs that are visible when the tab control is not multi-line.

User Object

Properties:

ID	ID of the user.
Name	Name of the user. (Last, First).
NameFL	Name of the user. (First Last).
Title	Title of the user.
Email	E-mail address of the user.
Phone	Phone number of the user.
DefaultSecCodeID	Default security code ID of the user.
PrimarySite	Primary site of the user.
Code	User code.
Enabled	Specifies whether the user is enabled and can log on.
UserType	Type of user. Can be one of the following values: utAdmin (0), utWorkGroup (1), utRemote (2), utWebOnly (3), utWebViewer (4), utConcurrent (5), utUnknown (6), utRetired (7), utTemplate (8).
ReportsTo	ID of the user's manager.
IsManager	Specifies whether the user is a manager.
DefectNotify	Specifies whether the user is notified when a defect is assigned to him or her.
SSApproval	Specifies whether the user can approve SpeedSearch items.
TicketNotify	Specifies whether the user is notified when a ticket is assigned to him or her.
TimeZone	Specifies the user's time zone.
Related Topic:	Application.Users

WorkArea Object

Properties:

Color	Specifies the background color of the work area.
Left	Indicates left coordinate of the list views client area covered by the work area.
Top	Indicates top coordinate of the list views client area covered by the work area.
Right	Indicates right coordinate of the list views client area covered by the work area.
Bottom	Indicates bottom coordinate of the list views client area covered by the work area.

WorkAreas Object

Properties:

Count	Returns the number of work areas in the list view.
Items	(Index) Provides indexed access to the WorkArea objects.

Functions:

Add	Creates and returns a new WorkArea object and adds it to the end of the Items property array.
Delete	(Index) Removes a WorkArea object from the Items property array.
Insert	(Index) Creates and returns a new WorkArea object and adds it to the Items property array in a specified position.

Chapter 3
Sage SalesLogix Stored Procedures

The following are Sage SalesLogix-specific stored procedures available with the Sage SalesLogix OLE DB Provider. They are 'Sage SalesLogix system' stored procedures that are executed in the same way as other stored procedures, but you will not see them in Microsoft SQL Server Enterprise Manager.

fx_rowaccess()

This is used in a SELECT query to determine the level of access the user has for each column, remembering that field level security will automatically NULL out the data, if the user doesn't have access *(so, the NULL value could be misinterpreted)*.

This function allows add-on applications to be a little smarter. A developer could dynamically set DB controls on a form to readonly=TRUE or enabled = FALSE or even visible=FALSE. They could also display a string inside the edit control, such as #secured#, etc.

They could also display a string inside the edit control, such as #secured#, etc.

Input parameters:	None
Resultset:	Returns an array of bytes where each indexed byte in the array represents the level of access for the corresponding column in the result set. The values for each byte are 0, 1, or 3 where: 0 - No field level access 1 - Read-only access 3 - Full read / write access

EXAMPLE

The following query, where the user has read access to the ACCOUNT field and no access to the CREDITAMOUNT field, yields the following result:

```
SELECT account, type, division, employees, credithold, creditamount, fx_rowaccess() from
ACCOUNT
```

ACCOUNT	TYPE	DIVISION	EMPLOYEES	CREDITHOLD	CREDITAMOUNT	FX_ROWACCESS
3Com	Prospect	Head Office	500	N	NULL	1333301
Abbott	Client	East Coast	800	Y	NULL	1333301

The same query, where the user has full access, yeilds the following:

ACCOUNT	TYPE	DIVISION	EMPLOYEES	CREDITHOLD	CREDITAMOUNT	FX_ROWACCESS
3Com	Prospect	Head Office	500	N	2,500,000	3333331
Abbott	Client	East Coast	800	Y	0	3333331

slx_ClearGlobalID

Clears the previous ID specified in slx_SetGlobalID, informing the logging system to determine synchronization requirements automatically.

Input parameters:	None
Resultset:	None

You must call slx_ClearGlobalID manually under the following circumstances:

- The Table you are updating does not have a direct relationship back to an Account.
- You wish to set the GlobalID to something other than an AccountID. For example, SecCodeID, UserID, or SiteCode.

The call to slx_SetGlobalID will override any existing logic for determining the GlobalID.

slx_CycleLogFile

Instructs the SLXLoggingServer.exe to close the current TEF file and open a new one. Usually occurs before a synchronization cycle.

Input parameters:	None
Resultset:	None

slx_DBIDs

Returns the requested number of Sage SalesLogix IDs for the specified table.

Input parameters:	param1 [string - table name]
	param2 [integer - number of IDs to return]
Resultset:	column1 [generated IDs (any number of rows)]

Slx_GetConcurrentAvailable

Returns the number of concurrent licenses available, up to the number requested.

Input parameters:	1 Alias
	2 Number
Resultset:	1 Concurrent available

slx_GetLoggedInServerInfo

Returns the port number and server name associated with a given alias.

Input parameters:	None
Resultset:	1 port (INT)
	2 server (CHAR 255)

slx_GetNativeConnInfo

Returns the full connection string to the underlying Microsoft SQL or Oracle database, without the SYSDBA password.

Input parameters:	None
Resultset:	column1 [string - full connection string as passed to the native provider (1 row)]

slx_GetServerList

Returns a list of the Sage SalesLogix Servers available on the network.

Input parameters:	One
Resultset:	Column1 [Server names (variable number of rows)]

slx_getUserInfo

Used for Windows Authentication, this stored procedure returns information about the user who is currently logged in. The first result is the Sage SalesLogix assigned ID. The second is the display name.

Input parameters:	None
Resultset:	1: USERID
	1: USERCODE

slx_RefreshDictionary

Refreshes the entire Data Dictionary cache. This function should only be used for troubleshooting. It may not exist in future Sage SalesLogix releases.

Input parameters:	None
Resultset:	None

slx_RefreshLogServer

Instructs the SLXLogingServer.exe to refresh the internal caches.

Input parameters:	None
Resultset:	None

slx_RefreshRWPass

Refreshes the password.

Input parameters:	None
Resultset:	None

slx_RefreshUser

Refreshes security information for the current user in the security cache. This includes password and other profile information. The refresh is then broadcast to the rest of the Sage SalesLogix clients.

Input parameters:	param1 [string - user name]
Resultset:	None

slx_RWPass

Returns the Read/Write password in an encrypted form. The SLXRWEL.dll COM object is used to decrypt this password so it can be included in the connection string for R/W access outside the Sage SalesLogix clients.

Input parameters:	None
Resultset:	param1 [string - R/W password]

slx_SendClientLog

Directly logs a log stream created on the Sage SalesLogix Client. For internal use only.

Input parameters:	1 Data (VARBINARY)
Resultset:	None

slx_SetDBPassword

Updates the System Administrator password on remote databases.

Input parameters:	UserID, OldPassword, NewPassword
Resultset:	None

slx_SetGlobalID

Informs the logging system that the following DML statements (INSERT / UPDATE / DELETE) are intended for the specified ACCOUNTID only. A valid USERID can be specified in place of an ACCOUNTID, indicating that intended changes are for a specified user.

For information on returning to the regular logging mode, see slx_ClearGlobalID.

Input parameters:	1: ACCOUNTID
Resultset:	None

You must call slx_SetGlobalID manually under the following circumstances:

- The Table you are updating does not have a direct relationship back to an Account.
- You wish to set the GlobalID to something other than an AccountID. For example, SecCodeID, UserID, or SiteCode.
- SQL2 type transactions contain key field and key value information and therefore the Logging Server can automatically determine the GlobalID.

The call to slx_SetGlobalID overrides any existing logic for determining the GlobalID.

slx_SetLogging

Specifies whether data or schema changes will generate sync log traffic for the current connection.

ON: Enables the generation of sync traffic for the current connection.

Off: Disables the generation of sync log traffic.

Input parameters:	param1: STATUS [string - either 'ON or 'OFF']
Resultset:	None

slx_settimezone

Sets the time zone of the corrent connection, which uses the same parameter values as the TIMEZONE extended property in the connection string. It is an INDEX value, as per the slx_timezonelist stored procedure. It only applies to the current connection.

Input parameters:	1:TIMEZONE
Resultset:	None

slx_TestLogPath

Returns 0 if the path exists on the server, otherwise it returns 1.

Input parameters:	1: PATH
Resultset:	1: RESULT (INT)

slx_timezonelist

Returns the list of time zones configured on the client machine. The INDEX or STANDARDNAME columns can be used with the TIMEZONE connection string property.

Input parameters:	None
Resultset:	1: INDEX
	2: STANDARDNAME
	3: LOCALNAME

slx_WNUpdateCAO

Generates a “What’s New” UPDATE transaction for the record with the primary key value in the ID field in the MAINTABLE table. FIELDNAME defines the field that has been changed, and the OLDVALUE / NEWVALUE parameters represent the old data and new data, respectively.

Input parameters:	1:MAINTABLE
	2:ID
	3:FIELDNAME
	4:OLDVALUE
	5:NEWVALUE
Resultset:	None

slx_WNInsertCAO

Generates a “What’s New” INSERT transaction for the record with primary key ID in table MAINTABLE.

Input parameters:	1:MAINTABLE
	2:ID
Resultset:	None

slx_WNDeleteCAO

Generates a “What’s New” DELETE transaction for the record with primary key ID in table MAINTABLE.

Input parameters:	1:MAINTABLE
	2:ID
Resultset:	None

slx_WNInsertAccount

Generates a “What’s New” INSERT transaction for a new account record. Parameter values match those within the following table.

ACCOUNTID	= ACCOUNTID
ACCTNAME	= ACCOUNT
CITY	= CITY
STATE	= STATE
ACCTMGR	= ACCOUNTMANAGERID
SECCODEID	= SECCODEID

Input parameters:	1:ACCOUNTID
	2:ACCTNAME
	3:CITY
	4:STATE
	5:ACCTMGR
	6:SECCODEID
Resultset:	None

slx_WNInsertContact

Generates a “What’s New” INSERT transaction for a new contact record. Parameter values match those within the table, as per the example in slx_WNInsertAccount.

Input parameters:	1:CONTACTID
	2:ACCOUNTID
	3:ACCTNAME
	4:CITY
	5:STATE
	6:ACCTMGR
	7:SECCODEID
	8:LASTNAME
	9:FIRSTNAME
Resultset:	None

slx_WNInsertOpp

Generates a “What’s New” INSERT transaction for a new opportunity record. Parameter values match those within the table, as per the example in slx_WNInsertAccount.

Input parameters:	1:OPPID
	2:DESCRIPTION
	3:ACCOUNTID
	4:ACCTNAME
	5:CITY
	6:STATE
	7:ACCTMGR
	8:SECCODEID
	9:POTENTIAL
	10:CLOSEDATE
Resultset:	None

sp_AliasList

Returns the list of available aliases on the server - this is a special stored procedure in that you can log on without an alias, call this function to get the alias, and then log on with the alias. All other stored procedures require that you log on with an alias before you can call them.

Input parameters:	None
Resultset:	column1 [list of aliases (any number of rows)]
Connection String:	Provider=SLXOLEDB.1;Data Source=(your Sage SalesLogix application server);Initial Catalog=SLXEVAL;User Id=admin;Password="";Persist Security Info=True;Extended Properties="Port=1706;Log=On

Data Source is the name you have given to your Sage SalesLogix application server.

Initial Catalog is the name set in the Connection Manager.

1706 is the default Port number. If your installation does not use the default port, enter the port number used.

Chapter 4
Sage SalesLogix OLE DB Provider

Connecting to the Sage SalesLogix Database

In Sage SalesLogix version 6.0 and later, database access and security is controlled by the Sage SalesLogix OLE DB provider. The OLE DB Provider allows ADO access to Sage SalesLogix data without the need for a proprietary API. Developers use standard ADO to access Sage SalesLogix data and the Sage SalesLogix OLE DB Provider automatically enforces user security and provides automatic data synchronization.

Technically, the Sage SalesLogix OLE DB Provider is a service provider. It extends the functionality of the SQL Server OLE DB (Data) Provider by adding three services which are not natively supported. These services are internal to the Sage SalesLogix OLE DB Provider and used automatically via the standard ADO interface. No additional or proprietary Sage SalesLogix ADO methods are required.

The Sage SalesLogix OLE DB Provider automatically:

- Performs data logging
- Manages user connections to the database and enforces licensing
- Enforces field- and record-level data security on all queries passed to the database
- Creates transaction exchange files (TEFs) for synchronization of data to remote users

Security no longer requires a server password on the user’s machine. All access rights are defined in the Administrator and on the Sage SalesLogix Server (the OLE DB Provider). The user only requires their log on password, and that is not cached or stored in the Registry.

The Sage SalesLogix OLE DB Provider is not limited to Sage SalesLogix Clients. Visual Basic, Application Service Provider (ASP) pages, Crystal Reports, or any other application that uses a standard OLE DB connection may access Sage SalesLogix data as a client. The interface is standard Microsoft ADO, requiring no proprietary API calls to access the data.

This section provides basic information about connecting to the Sage SalesLogix database.

Connection Strings

Basic Connection String

```
Provider=SLXOLEDB.1;Data Source=SalesLogix Server;Initial Catalog=SLXEVAL;User
 Id=admin;Password=””;Persist Security Info=True;Extended Properties=”Port=1706;Log=On
```

Properties

Property	Values	Required	Description
PROVIDER	SLXOLEDB.1	Yes	Specifies Provider name. Always SLXOLEDB.1.
DATA SOURCE	Host Name or IP Address	Yes	Specifies the host name or IP address of the Sage SalesLogix Server (SLXServer.exe).
INITIAL CATALOG	Valid ALIAS	Yes	A valid alias, as configured in the OLEDBConfigMgr.exe.
USER ID	Sage SalesLogix user name	Yes	Required for regular connection to view and manipulate data.
PASSWORD	Sage SalesLogix password	Yes	Required for regular connection to view and manipulate data.
PERSIST SECURITY INFO	TRUE or FALSE	Yes	When set to TRUE it will persist user authentication information. See ADO documentation for more details.
EXTENDED PROPERTIES	Any text string in double quotes	Yes	Please see following table for descriptions of extended properties.

Extended Properties

Property	Values	Required	Description
RWPASS	Read or read-write password	Yes	Possible scenarios for RW/RO passwords in OLEDBConfigMgr.exe: Only RW Password configured RWPASS must be present and contain valid R/W password to update data outside of the Sage SalesLogix clients. RW and RO password configured RWPASS must be present and contain valid R/O password to view data outside of Sage SalesLogix clients, however data cannot be updated. If RWPASS is present, then data can also be updated outside of the Sage SalesLogix Client.
PORT	Default: 1706 OR configured port	Yes	The port for which the SLXSystem.DLL and SLXServer.exe communicate on. It is configured in the SlxLocalServers.xml file for the SLXServer.exe.

Property	Values	Required	Description
LOG	OFF or ON		Specifies whether data or schema changes will generate sync log traffic. **ON** – enables the generation of sync traffic for the current connection**.** **OFF** – disables the generation of sync log traffic.
CASEINSENSITIVEFIND	OFF or ON		Overrides the default Sage SalesLogix provider behavior when using the IRowsetFind::FindNextRow or more common ADO Recordset::Find functionality. Selecting the check box toggles the CASEINSENSITIVEFIND "Extended Property" setting to 'On' (Sage SalesLogix Provider performs a case insensitive find). Clearing the check box toggles the CASEINSENSITIVEFIND "Extended Property" setting to 'Off' (default implementation for MDAC performs a case sensitive find).

Property	Values	Required	Description
TIMEZONE	[INDEX] or [STANDARDNAME]		Specifies the time zone to use when the OLE DB provider converts date / time fields between the database and client. Date time data is stored in Coordinated Universal Time (UTC) in the database, and must be converted to local time on the client. If this property is omitted, the provider will default to the current time zone settings of the client computer. The preferred method is using the numeric INDEX value, however as a user could edit the registry entries or create new time zones, this property may not be present or duplicated, so the option to use the STANDARDNAME is also provided as a back up. The STANDARDNAME is actually the registry key name[a] in the time zones section, and cannot be duplicated. If the INDEX or STANDARDNAME is not found (or in the case of INDEX, duplicated) a time zone not found error will be reported. To disable date/time conversion, specify NONE as the value. In a 6.2 system, this will typically mean all dates will be in GMT. For a list of INDEX and STANDARDNAME values, see the slx_timezonelist stored procedure.

Property	Values	Required	Description
TIMEZONE	[INDEX] or [STANDARDNAME]		Specifies the time zone to use when the OLE DB provider converts date / time fields between the database and client. Date time data is stored in Coordinated Universal Time (UTC) in the database, and must be converted to local time on the client. If this property is omitted, the provider will default to the current time zone settings of the client computer. The preferred method is using the numeric INDEX value, however as a user could edit the registry entries or create new time zones, this property may not be present or duplicated, so the option to use the STANDARDNAME is also provided as a back up. The STANDARDNAME is actually the registry key name[b] in the time zones section, and cannot be duplicated. If the INDEX or STANDARDNAME is not found (or in the case of INDEX, duplicated) a time zone not found error will be reported. To disable date/time conversion, specify NONE as the value. In a 6.2 system, this will typically mean all dates will be in GMT. For a list of INDEX and STANDARDNAME values, see the slx_timezonelist stored procedure.

Footnote

a. Under the Windows 2000, XP, and 2003 Server operating systems, the time zone information can be found in HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersions\Time_Zones.

b. Under the Windows 2000, XP, and 2003 Server operating systems, the time zone information can be found in HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersions\Time_Zones.

Extended Property Connection String Flags

The default value is shown in bold, meaning it is not necessary to include it for the default option.

Include Calculated Fields

Purpose: Calculated fields will be included in the result list for SELECT * queries if IncludeCalcFields is ON.

This functionality is used by reporting tools, such as Crystal Reports.

Syntax:

```
IncludeCalcFields=OFF|ON
```

Log

Purpose: Enables or disables logging for this connection.

Syntax:

```
Log=OFF|ON
```

Read/Write Password

Purpose: Enter a password in the Connection Manager in order to control write access to the database outside the Sage SalesLogix clients.

If the password field is blank, this property is not required, and all clients have read/write access to the Sage SalesLogix database using any tool.

Syntax:

```
RWPass=Read write password configured in OLE DB Configuration Manager.
```

Impersonate

Purpose: This property is available when logging on as the Admin user and instructs the provider to run under the context of the specified user's row and field level security.

Syntax:

```
Impersonate=SalesLogix username (e.g Lee, Pam, etc)
```

TrimCharFields

Purpose: This connection string property provides additional backward compatibility for connecting to legacy code. In earlier versions of Sage SalesLogix, the Borland Database Engine (BDE) trimmed strings coming from the database, but ADO does not. TrimCharFields allows trimming to be turned on when returning Data from the database on CHAR Data Types.

Syntax

```
TrimCharFields=Off|ON
```

Aliases

There two types of aliases:

Sage SalesLogix Aliases are created from the registry and are displayed in the Data Link Manager from the client machine, which in turn points to the Sage SalesLogix database through the provider.

External Aliases are created from the ADOAlias.ini file and/or within a script that uses the AddADOAlias() Procedure.

SLAPI has functions that are designed to login to a Sage SalesLogix database. These functions only look in the Sage SalesLogix Alias list created by the Data Link Manager.

Login(UserName,Password)

Loginto(Alias,UserName,Password)

Other functions, such as the following two, look in both the Sage SalesLogix Alias file and the External Alias file.

DBOpenSqlFromDatabase(Alias,SQLString,True)

DBOpenSqlFromDatabaseFor(Alias,UserName,Password,SQLString,True)

These functions locate the alias by first searching the Sage SalesLogix Alias list, then, if an entry is not found, searching the External Alias list. If no alias is found in either list, then 0 handle is returned and the logixerror set.

If a blank or Null alias is passed into the function, then the connection is set to the current connection of Sage SalesLogix.

If the alias is replaced with a fully formed ADO Connection string, then that ADO Connection string will be used and no search will take place.

For example:

```
strADOConn = "Driver=SQL Server;Server=Servername;
  Database=DatabaseName;UID=User1;PWD=Password1;"
    DBOpenSqlFromDatabase(strADOConn,SQLString,True)
```

The ADOAlias.ini File

The ADOAlias.ini file is located in the Sage SalesLogix folder and is created when Sage SalesLogix is loaded. Each alias entry is stored in the External Alias list.

Aliases in the ADOAlias.ini file have to be constructed in the following format:

```
[ADO Connection1]
Alias='Alias1'
ADOConnString='Driver=SQL
Server;Server=Servername;Database=Databasename;UID=User1;PWD=Password1;'
PassEncrypted='N'

[ADO Connection2]
Alias='Alias2'
ADOConnString='Driver=SQL
Server;Database=Databasename;UID=user2;Password=CA6885AB538DA444E444;Server=ServerName1;’
PassEncrypted='Y'
```

Use the encrypted option (ADO Connection2) if the ADO Connection string contains an encrypted password.

You can also create an External alias from within a Sage SalesLogix script by using the AddADOAlias() procedure.

When you use the AddADOAlias function within scripting, it does not add the new Aliases to the ADOAlias.ini file on the end user machine. Once aliases are loaded, they are memory resident and referenceable globally via scripting while the Sage SalesLogix application is active. These reference aliases are removed when the application is closed. Sage SalesLogix recommends writing a script, like the following example, that loads all needed ADO connection string aliases and attaching it to the OnOpen event of the Sage SalesLogix Client in the client options. This provides a central place to manage connection information.

Procedure: AddADOAlias (AliasName, ADOConnectionString,PassEncrypted)

For example:

```
strADOConn = "Driver=SQL Server;Server=Servername;Database=Databasename;UID=
  User3;PWD=Password3;"

    AddADOAlias "Alias3",strADOConn,"N"
    DBOpenSqlFromDatabase("Alias3",SQLString,True)
```

The previous procedure searches the External Alias list for an alias of the same name. If the alias exists, the procedure replaces it with the new connection string. If the alias is not found, the procedure adds the new alias to the list.

This method enables you to:

- Create the External Alias List dynamically without the ADOAlias.ini.
- Manage the alias list centrally.
- Have security without the need for encryption strings because the script requires an Admin login in the Architect.
- Change an alias connection string that already exists.

If the same alias name exists in both the Sage SalesLogix Alias list and the External list, the Sage SalesLogix Alias is used first.

Example Script

```
option explicit
sub main
Dim vHandle as Variant
Dim strSql as string
Dim Result as Variant
Dim strADOConn as string
Dim strAlias as string
Dim strPassword as String
```

Example 1: Create an ADO Connection String

```
strPassword = "Password3"
strADOConn = "Driver=SQL Server;"
strADOConn = strADOConn & "Database=DatabaseName;"
strADOConn = strADOConn & "UID=User3;"
strADOConn = strADOConn & "PWD="& strPassword & ";"
strADOConn = strADOConn & "Server=ServerName;"
strAlias = "Alias3"
```

Example 2: Add an External ADO Alias

```
AddADOAlias strAlias, strADOConn,"N"
```

Example 3: Connect with an Alias from the ADOAlias.ini file (External List)

```
vHandle = DBOpenTableFromDatabase("Alias1","Customers", True)
if LogixErrors Then
  msgBox "Handle:" & vHandle
  msgBox LogixErrorText
  exit sub
end if
msgBox DBGetValue(vHandle, "ContactName")
```

Example 4: Connect with an Alias from the AddADOAlias() (External List)

```
AddADOAlias strAlias, strADOConn,"N"

vHandle = DBOpenTableFromDatabaseFor("Alias3","user3","","Customers", True)
if LogixErrors Then
msgBox "Handle:" & vHandle
msgBox LogixErrorText
exit sub
```

```
end if

msgBox "For: " & DBGetValue(vHandle, "ContactName")
```

Example 5: Connect with an Alias from the Data Link Manager (Sage SalesLogix List)

' User name and Password is the Sage SalesLogix Username and Password

```
  strSQL ="Select * from Contact where Lastname ='Abbott'"
  vHandle = DBOpenSQLFromDatabasefor("SalesLogix_Eval","dan","",strSQL, True)
   if LogixErrors Then
     msgBox "Handle:" & vHandle
     msgBox LogixErrorText
     exit sub
  end if
  msgBox DBGetValue(vHandle, "Lastname")
```

Example 6: Connect with a Blank Alias (Current Connection)

' User name and Password is the Sage SalesLogix Username and Password but is ignored

strSQL ="Select * from Contact where Lastname ='Abbott'"

```
  vHandle = DBOpenSQLFromDatabasefor("","dan","",strSQL, True)
   if LogixErrors Then
     msgBox "Handle:" & vHandle
     msgBox LogixErrorText
     exit sub
  end if
  msgBox DBGetValue(vHandle, "Lastname")
```

Example 7: Connect with an ADO Connection String (Direct, No Alias)

When you use the AddADOAlias function within scripting, it does not add those new Aliases to the ADOAlias.ini file on the end users' machines. Once they are loaded, the aliases are memory resident and referenceable globally via scripting while the Sage SalesLogix application is active. These reference aliases are discarded when the application is closed. To have a central place to manage your connections, write a script that loads all needed ADO connection string aliases and attach it to the WhenOpen event of the Sage SalesLogix Client in the client options.

```
strPassword = "Password3"
strADOConn = "Driver=SQL Server;"
strADOConn = strADOConn & "Database=DatabaseName;"
strADOConn = strADOConn & "UID=User3;"
strADOConn = strADOConn & "PWD="& strPassword & ";"
strADOConn = strADOConn & "Server=ServerName;"
strAlias = "Alias3"
strSQL ="Select * from Customers where ContactName ='Ana Trujillo'"
 vHandle = DBOpenSQLFromDatabasefor(strADOConn,"SA","",strSQL, True)
  if LogixErrors Then
     msgBox "Handle:" & vHandle
     msgBox LogixErrorText
    exit sub
  end if
msgBox DBGetValue(vHandle, "ContactName")
dbClose(vHandle)
end sub
```

The Sage SalesLogix COM Interface

The Sage SalesLogix Component Object Model (COM) automation allows third-party applications to manipulate Sage SalesLogix clients during runtime. An application (written in VB, for example) can directly access Sage SalesLogix as long as the Sage SalesLogix Client is open.

All functions under the Application object model can be accessed using the object SalesLogix.SlxApplication. The following example initiates a conversation between Sage SalesLogix and third-party applications:

```
Sub Button1Click(Sender)
  Dim objSLXApp
  Dim objSLXBasicFunctions
  set objSLXApp = CreateObject("SalesLogix.SlxApplication")
  set objSLXBasicFunctions = objSLXApp.BasicFunctions

  msgbox objSLXBasicFunctions.CurrentUserID

  set objSLXBasicFunctions = nothing
  set objSLXApp = nothing
End Sub

Sub Button2Click(Sender)
  Dim objSLX
  set objSLX = CreateObject("SalesLogix.ClientObjix")

  msgbox objSLX.LogixErrorText

  set objSLX = nothing
End Sub
```

Example: Using the SLXApplication COM Interface

The following example shows using the SalesLogix.SLXApplication COM interface with DotNet and C#. To do this, you will need to add the SalesLogix Library as COM reference to your Visual Studio project.

1. Right-click on your Project Node in Solution Explorer. Select “Add Reference”.
2. On the COM tab, find SalesLogix Library with the Path to your SalesLogix.exe (C:\Program Files\SalesLogix.exe)
3. By adding the Sage SalesLogix namespace to your source file, you can now early bind to the Sage SalesLogix Application object.

The SLXApplication_Example.exe takes one command line parameter (ContactID). The application then opens a Contact Details MainView for that contact record by calling MainViews.add from the SLXApplication.LogonComplete event handler.

```
using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using SalesLogix;

namespace SLXApplication_Example
{
public class SLXApplication_Example : System.Windows.Forms.Form
{
  private SlxApplication FSLXApplication;
```

```
    private string FContactID;
    public SLXApplication_Example(string ContactID)
    {
      this.ClientSize = new Size(0, 0);
      this.WindowState = FormWindowState.Minimized;
      this.ShowInTaskbar = false;

      FContactID = ContactID;
      FSLXApplication = new SlxApplicationClass();
      // assign eventhandler for LogonComplete event
      FSLXApplication.LogonComplete += new ISlxApplicationEvents_LogonCompleteEventHandler
       (FSLXApplication_LogonComplete);
    }
    void FSLXApplication_LogonComplete()
    {
      // Display Contact Details mainview for ContactID record passed in the Command Line
      FSLXApplication.MainViews.AddEx("System:Contact Details",
       TxMainViewStyle.mvsMDIChild, true, true, FContactID, "");
    }

    static void Main(string[] args)
    {
      // pass in ContactID on command line
      string strContactID = "";
      if (args.Length > 0)
        strContactID = args[0];
      Application.Run(new SLXApplication_Example(strContactID));
    }

}
}
```

Sage SalesLogix Virtual Server Side Cursors

Virtual Server-Side Cursors (VSSC) are targeted at improving large group performance and memory usage in the Sage SalesLogix Network Client Groups / List views. Virtual server-side cursors can be used anywhere large, read-only datasets are being used. For small datasets, VSSCs can potentially degrade performance as client side cursors may be more efficient. For editable datasets, a client-side cursor is required.

The Sage SalesLogix Provider provides a server cursor implementation such that the client can request data on an as-needed basis. These are not true server-side cursors. In real server-side cursors, the process creates a cursor on the database server and keeps it open for the duration of the query. For Groups, this generally means the life of the application. If you have many Groups, you would have many server-side cursors active on the database server. Due to memory requirements of the permanent connection and numerous cursors active at any given time, database server performance and scalability will be affected. This benefit and liability was present in Sage SalesLogix v5.2.

On the client-side, server-side cursors benefit by providing pages of data continuously as you scroll back and forth through the result set. Client memory usage and initial results (assuming an efficient query) are significantly better in large Groups (thousands to millions of rows). The disadvantage is that scalability of the server is reduced because of the multiple client memory requirements on the server.

In the Sage SalesLogix Provider, VSSC is a read-only, bidirectional scrolling server-side cursor implementation. You will not be able to use an editable recordset to update the data, but you will be able to scroll forward and backwards through the data. Just like a server-side cursor, it requests pages of data from the server as needed. However, unlike server-side cursors, Sage SalesLogix will cache the data locally to save round trips if pages are revisited. This cache is dynamic in nature, so old pages will get thrown away to reclaim memory as necessary.

VSSCs will be available by a change to the adUseServer in the ADO connection. This means opening a new connection object as the standard Sage SalesLogix connection will be client-side.

Technically, VSSC is similar to the Microsoft client cursor engine (CCE) used for client-side recordsets (adUseClient in ADO).

If a user attempts to sort on a column that is not indexed, the query will execute slowly. The best approach is to prevent the user from sorting on non-indexed columns when using large groups that are in server-side mode. This is an admin-configurable feature which can be disabled if desired.

VSSCs significantly reduce the memory footprint on a client computer (tens of times depending on table size) and improve performance for large tables with thousands of rows. VSSCs always work in read-only mode. However, VSSCs should be avoided when using small tables because of overhead associated with caching and additional round-trips to the Microsoft SQL Server. VSSCs do not work with tables that do not have the SLX PRIMARY ID column.

What is a Cursor?

Operations in a relational database act on a complete set of rows. The set of rows returned by a SELECT statement consists of all the rows that satisfy the conditions in the WHERE clause of the statement. This complete set of rows returned by the statement is known as the result set. Applications, especially those that are interactive and online, cannot always work effectively with the entire result set as a unit. These applications need a mechanism to work with one row or a small block of rows at a time. Cursors are an extension to result sets that provide that mechanism.

A cursor keeps track of the position in the result set, and allows you to perform multiple operations row by row against a result set, with or without returning to the original table. Cursors conceptually return a result set based on tables within databases. With a server-side cursor, the server manages the result set using resources provided by the server computer.

Client Side Cursor

1. A query is executed by the client - for example; SELECT * FROM ACCOUNT ORDER BY ACCOUNT
2. The Sage SalesLogix OLE DB Provider opens a FORWARD ONLY, READ ONLY server side cursor for the underlying provider, and MDAC retrieves all the data into memory.

As an example, on the test hardware with a separate database server, a 2.7 million row dataset of the HISTORY table (SELECT * FROM HISTORY) consumed the 2GB limit for a Win32 process and took 5 minutes to run before crashing the process.

For small data-sets, this is a very efficient process, adding little overhead to the underlying provider.

Virtual Server Side Cursor

1. A query is executed by the client - for example, SELECT * FROM ACCOUNT ORDER BY ACCOUNT

2. The Sage SalesLogix OLE DB Provider invokes the VSSC engine (VSSCE) and creates a simple query to extract the unique row identifiers of the requested data. If a unique ID is not available, the query will fail with insufficient join or key information.
   - A unique ID is a standard 12 character, Sage SalesLogix ID field.
   - Performance HINT: for Microsoft SQL Server, ensure the use of a Clustered Index. Oracle may also benefit from a reverse-key index.
3. Assuming a unique ID was found, as in the example SELECT * FROM ACCOUNT ORDER BY ACCOUNT is executed on the database server and the resulting IDs are cached in a tightly packed data structure.

Note that the "ID" query honors all JOINS, WHERE cause criteria, and ORDER BY from the original query to ensure the same data set.

4. The VSSCE then loads a full page (currently defined as 100 rows) of data into the internal cache by issuing the original query with an IN clause containing the first 100 IDs added to the WHERE clause.
5. As the client scrolls through a recordset, or jumps to locations, the requested records fall on a certain page, and that page is automatically requested with the 100 IDs that fill that page.
6. The provider currently manages a cache of 10 pages of 100 rows, sorted by the most recently used. This means if a record is revisited, and the page still exists in memory, it will be moved back to the top of the list. Once 10 pages are cached, the last page is thrown out when new pages are requested. This paging process happens automatically and the client ADO recordset is completely unaware of this process. It only knows it has opened a server side cursor.
7. This paging process happens repeatedly as long as the ADO Recordset is open and connected (note that you must not disconnect the recordset by setting the ActiveConnection property to nil, nothing or null). Once the recordset is closed, the cache is thrown away. A cache is maintained for each open VSSC query.

As an example, on the same test hardware as the Client Side Cursor test, a 2.7 million row dataset of the HISTORY table (SELECT * FROM HISTORY) consumed about 36 MB of RAM for the ID cache and took about 4 seconds to run.

Usage in ADO

To start using VSSCs in ADO, you must first set the following three properties for the Recordset object:

```
CursorLocation = adUseServer
CursorType = ADOpenStatic
LockType = adLockReeadOnly
```

The following is a simple VBScript that opens a server side cursor and displays the first column of the first record.

```
'Create a Recordset object and a Connection object
Dim oADO
Dim oConnect
Dim sSQL
Dim sConnStr
'Create a Recordset object and a Connection object
Set oADO = CreateObject("ADODB.Recordset")
Set oConnect = CreateObject("ADODB.Connection")
sConnStr="Provider=SLXOLEDB.1;Password="""";" & _
"Persist Security Info=True;" & _
```

```
"User ID=admin;Data Source=XPJOHN2;" & _
"Extended Properties=""PORT=1706;LOG=ON"";Initial Catalog=SALESLOGIX_EVAL"
'Set the variable to the SQL statement that you would like to execute
sSQL = "select * from account where 1=1 order by type desc"
'Open the connection object
oConnect.ConnectionString = sConnStr
oConnect.Open
'Set the Recordset to the active connection and execute the SQL statement stored in sSQL
 variable
oADO.CursorLocation = 2 'adUseServer
oADO.CursorType = 1 'ADOpenStatic
oADO.LockType = 1 'adLockReadOnly
oADO.Open sSQL, oConnect
'Go to the first record in the result set and display the first column's value.
oADO.MoveFirst
Msgbox oADO.Fields(0)
'Clear the objects before ending the script
Set oADO = Nothing
Set oConnect = Nothing
```

Using RowsetFind interface.

ADO allows finding a record by a column value. VSSCs support the full set of conditions (see the Microsoft ADO reference for further information):

```
KEYCOLUMN=’primary-id’ (fastest)
ACOLUMN=’text’
ACOLUMN=INT
ACOLUMN=FLOAT
ACOLUMN LIKE ‘text%’
ACOLUMN LIKE ‘%text%’
ACOLUMN > INT
ACOLUMN <= FLOAT
```

The first condition executes faster than the others since the provider searches for a matching primary ID in the internal cache of IDs. All other conditions result in an additional round-trip to the Microsoft SQL Server to retrieve ACOLUMN data and perform a comparison on the client side. For this reason those conditions are slower.

Query / Relationship Requirements

The provider must be able to determine a unique single column key for each record. By this, we mean that if JOINs are included in the SQL statement, the provider must be able to determine a single key that uniquely identifies each row. This process relies on Global Joins being defined between the tables. Global Join data is used to hint the provider as to this relationship.

Left Joins, in most cases, will not be unique to a single column key and will not work. The exception is when a query contains a Left Join to a 1:1 table. In this case Sage SalesLogix can still use the primary table’s ID. A 1:1 table is defined in the JOINDATA table as having a SECONDARY field value of T.

The provider generates the following error codes when there is not enough information. (Note that the Sage SalesLogix default implementation in the Client uses this to fall back to the default cursor.)

IDS_E_SLXJOINTYPENOTSUPPORTED = $8004277F; (see Example 2)

```
IDS_E_SLXJOINCONDITIONNOTSUPPORTED = $80042780; (see Example 3)
IDS_E_SLXINSUFFICIENTJOINDATA = $80042781; (see Example 4)
```

Example 1: Correct

```
SELECT ... FROM ACCOUNT A INNER JOIN CONTACT C ON (A.ACCOUNTID = C.ACCOUNTID)
KEY: CONTACTID
```

The provider looks at the join data table, determines that the CONTACT table is a child table of the ACCOUNT table, and consequently uses the CONTACTID as the unique ID to handle paging.

Example 2:

IDS_E_SLXJOINTYPENOTSUPPORTED error returned

```
SELECT ... FROM ACCOUNT A
  OUTER JOIN CONTACT C ON (A.ACCOUNTID = C.ACCOUNTID)
```

The problem with this query is the unique table is CONTACT, however it is an Outer Join, so there is the potential for some contact records to not exist (even though this is invalid). Therefore, this could result in some NULL IDs.

Example 3:

IDS_E_SLXJOINCONDITIONNOTSUPPORTED error returned

This error occurs when additional join criteria is included in the ON clause, for example:

```
SELECT ... FROM ACCOUNT A
  OUTER JOIN CONTACT C ON (A.ACCOUNTID = C.ACCOUNTID AND TYPE='Customer')
```

To correct this use:

```
SELECT ... FROM ACCOUNT A
  OUTER JOIN CONTACT C ON (A.ACCOUNTID = C.ACCOUNTID) WHERE TYPE='Customer'
```

Example 4:

IDS_E_SLXINSUFFICIENTJOINDATA error returned

This error is reported when a global join between the tables in the FROM clause cannot be determined. To resolve this error add a global join between the parent and child tables.

It will also occur when multiple 1:M tables are included, as in the following example:

```
SELECT A1.ACCOUNT FROM ACCOUNT A1
  INNER JOIN CONTACT C1 ON (A1.ACCOUNTID = C1.ACCOUNTID)
  INNER JOIN OPPORTUNITY 01 ON (A1.ACCOUNTID = 01.ACCOUNTID)
```

The problem with this example is that both CONTACT and OPPORTUNITY are a direct 1:M child table of the ACCOUNT. We will see either duplicate CONTACTIDs or OPPORTUNITYIDs. We will also see either the joined contact or opportunity records duplicated many times. This is basically a Cartesian product query or Cross Join, where the two child tables are the participants.

For each account A that has both a contact C and opportunity O, where Cn is the number of contacts (Cn>0) and On is the number of opportunities (On>0), your result set will be:

- Each account will be duplicated Cn x On times.
- Each account will be duplicated On times.
- Each opportunity will be duplicated Cn times.

As the number of child rows in each of the 1:M tables grows, the number of A1.ACCOUNT values grows exponentially. The more realistic query, based on the data relationship is:

```
SELECT A1.ACCOUNT FROM ACCOUNT A1
  INNER JOIN OPPORTUNITY O1 ON (A1.ACCOUNTID = O1.ACCOUNTID)
  INNER JOIN OPPORTUNITY_CONTACT O2 ON (O1.OPPORTUNITYID = O2.OPPORTUNITYID)
  INNER JOIN CONTACT C1 ON (O2.CONTACTID = C1.CONTACTID)
KEY: OPPCONTACTID
```

Example 5: Limitations of using DISTINCT

For any given query, a fundamental requirement of the VSSC engine (VSSCE) is access to a single, unique Sage SalesLogix ID per row, sourced from one table. The process of obtaining which ID column to use is an iterative process, based on traversing the list of tables included in the query, to find the table with the most uniqueness.

Most uniqueness refers to the table that will provide a unique ID for each row in the result set.

Take the following query:

```
SELECT A.ACCOUNT, C.FIRSTMANE FROM ACCOUNT A
  INNER JOIN CONTACT C ON (A.ACCOUNTID = C.ACCOUNTID)
```

With the following data:

ACCOUNTID	ACCOUNT
A1	IBM
A2	Qantas

CONTACTID	ACCOUNTID	FIRSTNAME	LASTNAME
C1	A1	John	Bridges
C2	A1	John	Smith
C3	A1	Elle	Dolan
C4	A2	Pete	Jameson
C5	A2	Pete	Dover
C6	A2	Pete	North

The VSSCE will iterate through the tables, analyzing the JOINs and determine the CONTACT.CONTACTID is the most unique ID, since the CONTACT table is a 1:M of ACCOUNT. The algorithm is quite complex, relying on JOINDATA.

The result of the previous query would be:

ACCOUNT	FIRSTNAME
IBM	John
IBM	John
IBM	Elle
Qantas	Pete
Qantas	Pete
Qantas	Pete

There are a number of SQL constructs that almost always change the behavior of the result set, preventing a unique key from being obtained. These are aggregate, any GROUP BY and DISTINCT type queries.

Use the previous query and add a distinct clause as follows:

```
SELECT DISTINCT A.ACCOUNT, C.FIRSTNAME FROM ACCOUNT A
  INNER JOIN CONTACT C ON (A.ACCOUNTID = C.ACCOUNTID)
```

The result set is completely different:

ACCOUNT	FIRSTNAME
IBM	John
IBM	Elle
Qantas	Pete

If the VSSCE were to add a CONTACTID, it would change the result set, which violates a principal rule of the engine, whereby it is not allowed to change the query results in any way.

There are situations where the DISTINCT can be run in VSSC mode, when only fields from the parent tale and the primary table's ID column are included in the SELECT list. By default, the Sage SalesLogix Group Manager / Query Builder always add the primary table's ID column. So, the previous query would actually look like:

```
SELECT DISTINCT A.ACCOUNTID, A.ACCOUNT, C.FIRSTNAME FROM ACCOUNT A
  INNER JOIN CONTACT C ON (A.ACCOUNTID = C.ACCOUNTID)
```

If you remove the C.FIRSTNAME from the group query, the provider can execute in VSSC mode, since the ID is already in the result set.

VSSC in the Sage SalesLogix Client

Icons in the upper right corner of the List or Detail view (next to the Group Options button) indicate the current group mode as follows:

Icon	What Does it mean?
🗄	Group is in Virtual Server Mode - the group is being processed using Virtual Server-Side Cursors.
🗄🖥	Group is in Client Side Mode - the group is being processed on the client. It is not using Virtual Server-Side Cursors.

OLE DB Provider Extensions

When Sage SalesLogix is installed, the SLXOLEDBPLUGIN table is populated with five default records; an internal security record and four records that reference the Provider Extensions DLL. All five records are enabled by default. Use a query utility specific to your database to access the Sage SalesLogix tables.

Executing SELECT * FROM SLXOLEDBPLUGIN returns the following result set:.

COCLASS	DESCRIPTION	NAME	ENABLED	OBJECTTYPE
SLXSecurity	Sage SalesLogix base security object		T	I
SLXDefaults		SLXDefaults	T	D
SLXDateScalar		SLXDateScalar	T	S
SLXActivityBroker		SLXActivityBroker	T	E
SLXHistoryBroker		SLXHistoryBroker	T	E

SLXSecurity	Provides the default Sage SalesLogix security which limits visibility of data to users based on user security settings.
SLXDefaults	Inserts default values for table columns allowing for simplified queries.
SLXDateScalar	Enables the use of common functions across multiple database systems. The provider distinguishes between databases (Microsoft SQL and Oracle) and applies the appropriate parameters for each.
SLXActivityBroker	Extends basic security to activities.
SLXHistoryBroker	Extends basic security to history.

Customizing Sage SalesLogix Security

A custom security DLL can be used to extend or override the current row level security implementation in the Sage SalesLogix OLE DB Provider. To completely override the default Sage SalesLogix security, you must first disable the Sage SalesLogix base security object by changing the ENABLED column in the SLXOLEDBPLUGIN table to “F”. (Deleting this record WILL NOT remove the default security.)

- The Sage SalesLogix Premier license is required to create and install a custom security object.
- The Sage SalesLogix OLE DB Provider must be running on the same machine on which the custom security dll is installed.

To customize the Sage SalesLogix security object

Copy the following files from the Manifest folder, found on the Sage SalesLogix DVD, to your development folder. (All files must be in the same folder to successfully install a custom security object.)

- SLXProviderPlugin.exe
- SLXProviderPlugin.dll

Development

1. Design your security customization and build a ‘CustomSecurity’.dll (you can use the SLXProviderExtensions.dll as an example).
2. Copy the ‘CustomSecurity’.dll to your development folder.
3. Create a ‘Custom_Manifest’.xml file.

Example .xml

```
<?xml version="1.0" encoding="utf-8" ?>

<slxplugin name="PROVIDEREXTENSIONS" file="SLXProviderExtensions.dll" uid="SYST0000000A">

<defaults coclass="SLXDefaults" enabled="true" continueonfail="true" uid="SYST0000000A">

    <table name="*" field="MODIFYDATE" type="all" />

    <table name="*" field="MODIFYUSER" type="all" />

    <table name="*" field="CREATEDATE" type="insert" />

    <table name="*" field="CREATEUSER" type="insert" />

</defaults>

<scalarfunction coclass="SLXDateScalar" enabled="true" continueonfail="true"
uid="SYST0000000B”>
```

```
    <function name="SLXDATEPART" />

    <function name="SLXDATENAME" />

    <function name="SLXDATEDIFF" />

    <function name="SLXDATEADD" />

    <function name="SLXGETDATE" />

    <function name="SLXSTR" />

    <function name="SLXCAT" />

</scalarfunction>

<securityobject coclass="SLXActivityBroker" enabled="true" continueonfail="true"
uid="SYST0000000C"/>

</slxplugin>
```

Installation

1. Create a 'Custom'.UDL file indicating the Sage SalesLogix OLE DB Provider as your data source.
2. Run "slxproviderplugin.exe -m 'Custom_Manifest'.xml -u 'Custom'.udl" from the DOS command prompt. (See description under "Format of the configuration file" on page 171.)
3. Verify the installation was successful.
4. Close and restart all applications which connect through the Sage SalesLogix OLE DB Provider. (This will ensure the most recent DLLs are being read.)

The security class is derived from the ISLXSecurityBroker interface.

There are three methods implemented:

1. **Initialize** (pSecurityInit)
   Stores the SLXSecurityInit parameter in a member variable. Can also be used for initialization of other class members.
2. **Uninitialize**
   This method is not currently used by Sage SalesLogix in this version but can be used for class cleanup.
3. **GetSecurityObject** (pSQLQuery: ISLXSQLQuery)
   Creates a security class type depending on the type of query.
   Creates one of the following instances of classes:
   - TSLXSelectSecurity
   - TSLXUpdateSecurity
   - TSLXInsertSecurity
   - TSLXDeleteSecurity

TSLXSelectSecurity

The TSLXSelectSecurity class is derived from the ISLXQuerySecurity parent interface and implements one method.

Secure - creates QRYSECRESULT

- Does not modify security for Admin
- Analyses JOINS in a loop
- If related to activity or history table then modifies the WHERE clause.
  Adds a new condition to check the security permission of the user and filters the data retrieved appropriately.

Secure Method returns:

QRYSEC_OK - Security applied successfully.
QRYSEC_OKWITHROW - Reserved for future use.
QRYSEC_NOEXECUTE - User does not have rights to execute the query.

Default Values

The Sage SalesLogix Defaults technology inserts default values for table columns allowing for simplified queries. TPROVIDERDEFAULTS is derived from the ISLXDEFAULT Interface.

Methods

1. **Initialize**
   Called once at object creation. Single IN parameter SLXCONNECTIONINFO
   - CONNECTIONSTRING (DB)
   - SLXCONNECTIONSTRING (SLXOLEDB)
   - SLXDBTYPE (database type)
2. **Uninitialize**
   Used to clean up code.
3. **SetDefaults**
   - ISLXSQLQuery (IN)
     Parse tree
   - Columnname (IN)
     Name of column default to supply.
     Table available from ISLXSSQLQuery.
   - UserID (IN) (User executing query).

If insert query does not specify values for columns:

Modify User

Create User

Modify Date

Create Date

SECCODEID

If update query does not specify values for the the columns:

Modify User

Modify Date

Then the method extends the query to insert the proper (default) values.

SECCODEID is inserted only for child tables of secured tables used.

Default = ‘SYST000000001’

Scalar Functions

The Sage SalesLogix Scalar technology enables the use of common functions across multiple database systems. These functions are generally expressed as SQL functions with the SLX prefix. The provider distinguishes between databases (Microsoft SQL and Oracle) and applies the appropriate parameters for each.

Primary: Abstract database specific functions

- Write single SQL for any database
- Date functions
  a. SLXDATEPART
  b. SLXDATENAME
  c. SLXDATEDIFF
  d. SLXDATEADD
  e. SLXGETDATE

Single implementation can handle many scalar functions

- FunctionName is passed as a parameter at query execution

Additional Functions

SLXSTR	Converts a number or a date to a string. Microsoft SQL Server uses STR function, Oracle – TO_CHAR.
SLXCAT	SLXCAT - puts two or more strings together to make one. Microsoft SQL Server uses '+' operator, Oracle – '||'

Methods

1. **Initialize**
   Called once at object creation.
   Single IN parameter SLXCONNECTIONINFO
   - CONNECTIONSTRING (DB)
   - SLXCONNECTIONSTRING (SLXOLEDB)
   - SLXDBTYPE (database type)
2. **Uninitialize**
   Used to clean up code
3. **GetScalarReplacement**
   - ISLXSQLQuery (IN)
     Parse tree
   - FunctionName (IN)
     Name of scalar function being called
   - ISQLParamValues (IN)
     Variable list of parameters passed to scalar function
   - SQLExpression Text (OUT)
     Valid SQL expression string to be injected in query
   - SLXSTR
     Converts a number or a date to a string.
     Microsoft SQL Server uses STR function, Oracle – TO_CHAR.
   - SLXCAT
     Puts two or more strings together to make one.
     Microsoft SQL Server uses '+' operator, Oracle – '||'

Format of the configuration file

Slxplugin Root of the XML document.

Attributes:
Name – any user friendly name that refers to the extension;
File – a DLL file with the extension code;
Uid – optional parameter that can be used to set primary key value for the inserted row into SLXOLEDBPLUGIN table; can be used to locate the inserted row; all values starting with 'SYST' characters are reserved for internal use;

Defaults Configuration for the "Defaults" functions. These functions allow setting fields default values when INSERT or UPDATE is executed.

Attributes:
Coclass – COM class name;
Enabled – Boolean value (T or F) that enables the function;
Continueonfail – Boolean value (T or F) that allows Provider to execute statement even if the function is failed;
Uid – optional parameter that can be used to set primary key value for the inserted row into SLXSQLDEFAULT table; can be used to locate the inserted row; all values starting with 'SYST' characters are reserved for internal use;

Table Table where the default value should be inserted.

Attributes:
Name – table name; use * for any name;
Field – field name in a table;
Type – type of the function (insert, update or all); the type define which query type (INSERT, UPDATE or both) can use the function;

Scalarfunction Configuration for the "Scalar" functions. These functions allow replacement of original query pieces for better inter-database compatibility and other purposes; in future Provider versions the functions could be implemented as real scalar functions;

Attributes:
Coclass – COM class name;
Enabled – Boolean value (T or F) that enables the function;
Continueonfail – Boolean value (T or F) that allows Provider to execute statement even if the function is failed;
Uid – optional parameter that can be used to set primary key value for the inserted row into SLXSQLDEFAULT table; can be used to locate the inserted row; all values starting with 'SYST' characters are reserved for internal use;

Function Function name; the name is recognized by Provider and the extensions DLL called for the text replacement;

Attributes:
Name – function name;

Securityobject Configuration for the external security object. The security object allows custom securing of queries.

Attributes:
Coclass – COM class name;
Enabled – Boolean value (T or F) that enables the function;
Continueonfail – Boolean value (T or F) that allows Provider to execute statement even if the function is failed;
Uid – optional parameter that can be used to set primary key value for the inserted row into SLXSQLDEFAULT table; can be used to locate the inserted row; all values starting with 'SYST' characters are reserved for internal use;

Auto-Increment Support for Primary Keys

Auto-Increment Support for Primary Keys functions in the same way as auto-increment (identity) columns work in Microsoft SQL Server. When a new record is added to a recordset, but the primary key value is omitted, the OLE DB Provider will automatically generate the key value. If ADO is being used, it will subsequently populate the primary key in the recordset after calling the Update or Update Batch method. This functionality is achieved with support for the @@IDENTITY variable in the Sage SalesLogix OLE DB Provider.

The @@IDENTITY variable works the same as in SQL Server, where it represents the last auto-increment value executed on this connection. It is possible to manually execute a statement like "SELECT * FROM ACCOUNT WHERE ACCOUNTID = @@IDENTITY" to retrieve the last row that was inserted with an auto-incremented key.

This functionality can be used in ADO tools such as ADO Explorer. Select an editable recordset and specify values for all the necessary columns in the editable grid except the primary key. As you move to the next row, to post the inserted record to the database, the primary key value will be displayed in the primary key column, assuming it was included in the SELECT statement. It is not a requirement to include the primary key in the SELECT.

Metadata in the form of a column called AUTOINCREMENT is displayed in the SECTABLEDEFS system table. It should contain either "T", "F" or NULL and is case-sensitive. "T" is only supported for the PRIMARY KEY and is therefore ignored in other columns. A value other than "T" is treated as "F".

This is displayed as a check box for the key column within the DB Manager.

Key Generation

The Sage SalesLogix OLE DB Provider detects when OTHER keys are exhausted and begins a new sequence. For example, where XXXX is a site-code the key sequence may look like the following:

QXXXXA000234

QXXXXA000235

QXXXXA000236

QXXXXZZZZZZX

QXXXXZZZZZZY

QXXXXZZZZZZZ

RXXXX0000000

RXXXX0000001

RXXXX0000002

RXXXXZZZZZZX

RXXXXZZZZZZY

RXXXXZZZZZZZ

SXXXX0000000

SXXXX0000001

SXXXX0000002

SXXXXZZZZZZX

SXXXXZZZZZZY

SXXXXZZZZZZZ

TXXXX0000000

TXXXX0000001

TXXXX0000002

ZXXXXZZZZZZX

ZXXXZZZZZZY

ZXXXXZZZZZZZ

0XXXX0000000

0XXXX0000001

0XXXX0000002

SLX_DBIDS('table', keyCount) may return less than requested keyCount in one specific situation – when keys overflow occur. However, it always returns at least 1 key.

Error Messages

The following table is a list of server generated error messages.

Error Message	Component	Comments
BeginTrans-Transaction Level xxx	SLXLoggingObj2.DLL	An internal failure attempting to begin a QUEUE file transaction. Possibly an out of memory error.
CheckNeedToCycle - Error cycling logfiles: xxx	SLXLoggingServer.exe	While attempting to cycle the log files, an exception was raised. Additional information may assist in troubleshooting this error.
CommitTrans-Transaction Level xxx	SLXLoggingObj2.DLL	An internal failure attempting to commit a QUEUE file transaction. This is likely to be a file IO error as the files are moved and renamed from *.trn to *.que. Additional information may assist in troubleshooting this error.
Error determining GlobalID: xxx Transaction will be sent to all users.	SLXLoggingServer.exe	The message "xxx" indicates the reason for the warning. The transaction will still be logged and synchronized, however as the ACCOUNTID could not be determined, it will be transmitted to all remote users.
Error getting sync info from database: Provider=%s;Password=***** ;Persist Security Infor=True;User Id=%s;Initial Calalog=%s;Data Source=%s. ERROR: xxx	SLXLoggingServer.exe	An attempt was made to read the SYNCSERVER table from the database, however the operation failed. The connection information or additional error information will assist in troubleshooting this error.
Error getting system info from database: xxx	SLXLoggingServer.exe	A failure occurred attempting to read the SYSTEMINFO table, where the SYSTEMINFOID is "PRIMARY". Additional information may assist in troubleshooting this error.
Error initializing connection pool: xxx	SLXLoggingServer.exe	An attempt was made to create an instance of the Microsoft OLE DB Simple Provider (MSDAOSP.1) to enable OLEDB connection pooling. As this provider should always be present for a valid MDAC installation, there may be issues with the system.
Error initializing queue file list: xxx	SLXLoggingServer.exe	The Logging server was attempting to retrieve a list of files in the "…All Users\ApplicationData\SalesLogix\Syn c\QUEUEFiles" folder but failed. Additional information may assist in troubleshooting this error.
Error initializing worker thread: xxx	SLXLoggingServer.exe	Additional information may assist in troubleshooting this error.

Error Message	Component	Comments
Error Processing Queue file xxx : xxx	SLXLoggingServer.exe	The failed queue file will be moved to the QUEUEFiles\Failed folder. Additional information may assist in troubleshooting this error.
Error searching for Queue files: xxx	SLXLoggingServer.exe	While processing queue files, the logging server will search for any further generated files before sleeping momentarily and yielding CPU time. Additional information may assist in troubleshooting this error.
Error writing to Logfile: WriteToLog – xxx	SLXLoggingServer.exe	Failure to write to the log file, which could occur for many reasons. For example, the disk is full. Additional information may assist in troubleshooting this error.
RollBackTrans-Transaction Level xxx	SLXLoggingObj2.DLL	An internal failure attempting to commit a QUEUE file transaction. This is likely to be a file IO error as an attempt was made to delete the queue files making up the transaction. Additional information may assist in troubleshooting this error.
SetAHCFlags - Error determining Activity, History, or Calendar flags for Table: xxx Key: xxx	SLXLoggingServer.exe	An internal error.
SyncServerInfo-LoadFromQueries: No Records returned for query (xxx)	SLXLoggingServer.exe	There are no records in the SYNCSERVER table.
TLoggingFile - Error Cycling log file:xxx	SLXLoggingServer.exe	An attempt was made to close the log file in question however it may have been deleted or is locked by another process.
Unable to load non-synching tables.	SLXLoggingServer.exe	An exception occurred trying to read the RESYNCTABLEDEFS table. Perhaps the database server was shut down.
UserCodeToUserID - Error retrieving userid: xxx	SLXLoggingServer.exe	Failure to determine the USERID for the given user name as an attempt was made to read the USERSECURITY table. As this is an exception, it is likely to be an event such as the DB server shutting down.

Global / UTC date/time support

UTC or Coordinated Universal Time support is implemented by the provider, to simplify the handling of dates when accessing data from the database. In this section GMT or UTC may be used interchangeably.

At the database level, all date/times are stored in UTC. Additionally, all date/times that are passed through the Sage SalesLogix system, such as What's New transactions or TEF files, will be in UTC. This assumption allows the system to make date/time comparisons without the need for conversion.

When a connection is established through the provider, the default is to use the time zone settings of the client machine, as per the Date Time control panel applet. However, if a valid TIMEZONE connection string property is present, the current connection will use these overridden options.

Date/time data is retrieved from the database and converted on the client machine to local time based on the time zone rules previously stated. Daylight savings is taken into consideration for each date analyzed, as dictated by the time zone in the Microsoft Windows registry.

Expectations

When using the provider to take advantage of UTC support, there are a few expectations of the SQL and data.

- Conversions are applied to any column of a SELECT statement that results in a date / time data type.
- Conversions are only applied to a date/time parameter value of a parameterized query or an ISO formatted date/time literal string when used in a WHERE clause or as any part of an INSERT, UPDATE and DELETE.

  An ISO date takes the format of 'CCYYMMDD hh:mm:ss' where:

  CC - Century

  YY - Year

  MM - Month

  DD - Day

  hh - 24 Hour

  mm - Minute

  ss - Second

Do not use date / time scalar functions. As an example, the Microsoft SQL Server GetDate() function result will not be converted.

It is important to note that in a mixed Oracle (host) / Microsoft SQL Server (remote) environment, attempting to use scalar functions in INSERT, UPDATE and DELETE statements will not synchronize correctly if the scalar function is specific to a certain vendor. This SQL will be synced as a literal SQL statement, and fail to execute on the destination database server.

Non-UTC date/time fields

It is still anticipated that some fields will not require date/time conversion, such as date only or time only type fields. To accomplish this, the SECTABLEDEFs table has a new column named DATETIMETYPE. This is a single character field that makes the following assumptions:

- If it contains a 'D' or 'T' (date only or time only respectively) it indicates to the OLE DB provider that this field does not require UTC conversion, and will therefore be passed through without alteration.
- 'Z' is reserved, is unsupported by third party use, and is not guaranteed to be supported in future versions of Sage SalesLogix.

- 'U' (or any other value) indicates the default, and the date/time field is stored as UTC, and should be converted to the local time on the client machine.

The DB Manager will set these values appropriately when creating or altering tables with date/time fields. The user can specify a date only, time only or regular date/time field.

Synchronization

Synchronization has been implemented in the OLE DB provider, to leverage the skills of developers already familiar with ADO data access components. In addition there are many third-party applications and development environments that can take advantage of ADO as a data access mechanism. A user need only establish an ADO connection, using the Sage SalesLogix OLE DB provider, and all subsequent insert, update or delete transactions will be automatically logged for synchronization, including DDL such as CREATE TABLE or ALTER TABLE.

This section describes how the provider makes decisions of what type of transaction to log, based on the incoming SQL query, and the advantages or limitations the different transactions have when processed by the synchronization system.

This section also provides some guidelines on how to best write SQL (if it is deemed necessary) so as to best take advantage of the synchronization engine's features.

Transaction Types

There are a number of different transaction types supported by the Sage SalesLogix synchronization system, however only the types generated by the provider will be discussed here. The transaction type is the name as it appears in the Sage SalesLogix TrnViewer application.

Transaction Update2

Description Generated by a simple UPDATE statement that

- contains only simple expressions,
- for every new value in the SET clause, a corresponding old value exists in the WHERE clause and
- contains the primary key in the WHERE clause, consequently resulting in only 1 row being modified.

If an ADO recordset is edited, ADO will always generate an UPDATE query in this format, but literal values will be replaced by parameters, generating a parameterized query.

An example of a query containing literal values.

```
UPDATE ACCOUNT
  SET ACCOUNT = 'Sydney Opera House', TYPE = 'Customer'
WHERE
  ACCOUNTID = 'AQF8A00001BC' AND ACCOUNT = 'Opera House' AND TYPE =
'Prospect A'
```

The same query with parameters, as generated by editing an ADO recordset

```
UPDATE ACCOUNT
  SET ACCOUNT = ?, TYPE = ?
WHERE
  ACCOUNTID = ? AND ACCOUNT = ? AND TYPE = ?
```

Conflict resolution occurs at the field level, since an Update2 is broken into individual field updates. As an example, the previous statement would be executed on the 'destination database' where the TEF is applied as two statements:

```
UPDATE ACCOUNT
  SET ACCOUNT = 'Sydney Opera House'
WHERE
  ACCOUNTID = 'AQF8A00001BC' AND ACCOUNT = 'Opera House'

UPDATE ACCOUNT
  SET TYPE = 'Customer'
WHERE
  ACCOUNTID = 'AQF8A00001BC' AND ACCOUNT TYPE = 'Prospect A'
```

If (as an example) the second of these fails to update the row, conflict resolution will examine the rules, and if it is deemed the update should be applied (e.g. Remote always wins), then the update will be executed as

```
UPDATE ACCOUNT
  SET TYPE = 'Customer'
WHERE
  ACCOUNTID = 'AQF8A00001BC'
```

Pros	The synchronization engine is able to apply conflict resolution rules to an Update2, given the presence of old and new values in the UPDATE. This is one of the single most important reasons to format queries in this manner or use ADO recordsets.
Cons	

Transaction	SQL2
Description	This is generated by any supported DDL (e.g. CREATE TABLE), an INSERT or DELETE statement and complex UPDATE statements or those that do not contain 1 old value for each corresponding new value.
Pros	
Cons	When synchronizing UPDATE statements via SQL2 transactions, no conflict resolution occurs, so there is no guarantee that this data will apply correctly. This is typically a concern when more complicated WHERE clauses are used.

Sage SalesLogix Program Components

Sage SalesLogix OLE DB Provider components (client only)

Component	Description
SLXOLEDB.DLL	COM component. Primary Sage SalesLogix OLE DB Provider interface library. See Microsoft OLEDB documentation for further details of OLE DB technology.
SLXDBEngine.DLL	COM component. The SLXDBEngineDLL is the core Sage SalesLogix DB engine, handling parsing, business logic and processing of SQL queries. It is used by the SLXOLEDB libary only.
SLXSystem.DLL	A singleton application that channels multiple requests from the SLXSystemDLL, which is invoked from the client process, to the Sage SalesLogix Application Server. SLXSystem.exe is also responsible for transferring .QTS files to the Sage SalesLogix Server where they become .QUE files.
SLXLoggingObj2.DLL	COM component. Creates and persists .QTS files on client to later be processed by SLXLoggingServer and converted to TEFs. Used solely by the Sage SalesLogix OLE DB provider.
SLXSL.DLL	Regular DLL. Low-level socket libary.
SLXRWEL.DLL	COM component. Sage SalesLogix Read/Write encryption library. This component is used to encrypt and decrypt security sensitive information that is passed between DLLs or across the network between Sage SalesLogix components. In addition, third party applications can use this library to decrypt access passwords such as the Read/Write password.

Sage SalesLogix Application Server components (server only)

Component	Description
SLXServer.exe	Handles requests from the client SLXSystem component, such as database connection information. Other services, that make sense to be centralized, are located with this component.
SLXLoggingServer.exe	Processes .QUE files on server to generate Transaction Exchange Files (TEFs) for Sage SalesLogix synchronization.
SLXLicenseMgr.DLL	COM component. Server component to manage Sage SalesLogix licensing.
OLEDBConfigMgr.DLL	COM component. Server component to manage Sage SalesLogix aliases, configured by the Sage SalesLogix connection manager.
SLXSL.DLL	Regular DLL. Low-level socket library.

Component	Description
SLXRWEL.DLL	COM component. Sage SalesLogix Read/Write encryption library. This component is used to encrypt and decrypt security sensitive information that is passed between DLLs or across the network between Sage SalesLogix components. In addition, third party applications can use this library to decrypt access passwords such as the Read/Write password.

Appendix A
Database Type Definitions

This section provides definitions for non-English column values found in tables within the Sage SalesLogix database.

Values for USERSECURITY.TYPE

Value	Definitions
M	Remotes
N	Network
C	Concurrent User
T	Web Only
V	WebViewer
R	Retired
T	Template
A	Add On

Values for INDEXDEFINITION.TYPE

Value	Definition
0	File System
1	Database

Values for INDEXDEFINITION.USERACCESS

Value	Definition
0	Customer
1	Internal

Values for CAMPAIGNTASK.OWNERTYPE

Value	Definition
0	Existing Users/Teams
1	Department
2	Contact
3	Other Individual
4	None

Values for PLUGIN.TYPE

Value	Definition
0	Processes
2	Scripts (Basic)
3	Views (Legacy)
5	Scripts (SQL)
8	Groups
13	Macros
14	Strips (Menu)
15	Strips (Toolbar)
18	Bitmaps
19	Crystal Reports
23	Groups
25	Templates
26	XML
27	Reports

Values for RESYNCTABLEDEFS.OMNIDIRECTIONAL

Value	Definition
T	Sync to and from Remotes
F	Syncs from Host to Remoes only
X	Does not sync

Values for SECCODE.SECCODETYPE

Value	Definition
U	Users and Templates
G	Team
D	Department
S	System

Values for SUBSCRIPTIONRULES.STATUS

Value	Definition
S	Subscribe
U	Unsubscribe
X	Deleted / converted to unsubscribe

Values for SYNCFILETRACKING.FILESTATUS

Value	Definition
0	Requested
1	Not Found
2	Received

Values for ACTIVITY.TYPE

Value	Definition
262146	Phone Call
262145	Meeting
262147	To Do
262162	Personal Activity

Events are saved in Event table.

Lit Requests are saved in Litrequest table.

Values for TICKET table picklists (dynamic)

Value	SQL Statement
Ticket Activity	select text, itemid from sysdba.picklist where picklistid = 'kSYST0000372'
Ticket Activity Public Access	select text, itemid from sysdba.picklist where picklistid = 'kSYST0000373'
Ticket Area	select text, itemid from sysdba.picklist where picklistid = 'kSYST0000374'
Ticket Status	select text, itemid from sysdba.picklist where picklistid = 'kSYST0000375'

Values for RMA table Picklists (dynamic)

Value	SQL Statement
Return Priority	select text, itemid from sysdba.picklist where picklistid = 'kSYST0000410'
Return Status	select text, itemid from sysdba.picklist where picklistid = 'kSYST0000411'
Return Type	select text, itemid from sysdba.picklist where picklistid = 'kSYST0000412'

Defects

To get a list of Types relating to Defects (these lists are dynamic, but the ID should be the same for all databases).

Value	SQL Statement
Defect Activity	select text, itemid from sysdba.picklist where picklistid = 'kSYST0000336'
Defect Area	select text, itemid from sysdba.picklist where picklistid = 'kSYST0000337'
Defect Fixed in Build	select text, itemid from sysdba.picklist where picklistid = 'kSYST0000338'
Defect Priority	select text, itemid from sysdba.picklist where picklistid = 'kSYST0000339'
Defect Resolution	select text, itemid from sysdba.picklist where picklistid = 'kSYST0000340'
Defect Severity	select text, itemid from sysdba.picklist where picklistid = 'kSYST0000341'
Defect Status	select text, itemid from sysdba.picklist where picklistid = 'kSYST0000342'
Defect Type	select text, itemid from sysdba.picklist where picklistid = 'kSYST0000343'

Value	SQL Statement
Defect Urgency	select text, itemid from sysdba.picklist where picklistid = 'kSYST0000344'
Defect Frequency	select text, itemid from sysdba.picklist where picklistid = 'kSYST0000406'
Defect Source	select text, itemid from sysdba.picklist where picklistid = 'kSYST0000407'
Defect Target Version	select text, itemid from sysdba.picklist where picklistid = 'kSYST0000408'
Defect Version Found	select text, itemid from sysdba.picklist where picklistid = 'kSYST0000409'

Glossary

Account A company with which there is a current business relationship. Accounts are owned by an individual user, a team of users, Everyone (view-only), or Everyone.

ActiveX® Data Objects (ADO) The interface used to access data sources. ADO contains drivers that allow support system components to access and manipulate data in a database server through any OLE Database provider.

Activity Work recorded on a Call Ticket that was performed to solve the customer's problem. In some cases, the time or dollar amount of the activity is deducted from the customer's contract.

Administrator 1. The component used to configure and manage Sage SalesLogix. 2. A person with full rights in Sage SalesLogix. He or she can add or remove other users and has access to the Administrator. The administrator can also customize many of the fields and screens used throughout Sage SalesLogix.

Architect A development environment for creating customized views, menus, toolbars, and scripts for Sage SalesLogix.

Base Table A table from which a view is derived. Also called an underlying table. A view can have one or more base tables or base views. A base table does not depend on any other table; its description and data are physically stored in the database.

Blob Field Another term for memo field. Any field in which you can type and save up to 2 gigabytes of text, such as the Problem or Result field or Notes tab. These fields also usually support the use of [F9] key to date-/time-/name-stamp the field.

Bundle A bundle is a group of plugins or other customizable components that are packaged together for installation as a unit, rather than having to move them one by one, or re-create them on every database. While bundles are actually created in the Architect, web elements such as actions, aliases, queries, and templates must be tagged as a Web Bundle in order to facilitate inclusion in a bundle. All HTML files in a Web Bundle are saved in a folder under the HTML directory with the bundle name. Bundles are installed using the Bundle Manager in the Administrator.

Category A topic used to describe the nature of a customer's problem. This may simply be a name of one of your company's products, or it may be a short phrase, defined by the administrator, that describes a common question or problem experienced by customers.

Company A record for a customer, whether the customer is a private individual or an organization, that owns a supported product and/or support contract. The people at that company (one or more) are listed as Contacts on the Company record.

Concatenation Combining two or more character strings or expressions into a single character string or expression, or combining two or more binary strings or expressions into a single binary string or expression.

Concurrent Users The number of users accessing the database at any one time. The number of concurrent users impacts the relational database management system (RDBMS) and server hardware you choose.

Contacts The individuals in a company (account) with whom users interact. Each account can contain one or more contacts.

Contract An agreement between your organization and the customer to provide support services, either for free or for a specified fee. In some cases, the contract is "billed" for the time or dollar amount recorded in activities on tickets for the customer. The contract may expire when a set amount of hours or dollars is reached, or when a certain date is passed.

CRLF A carriage return (CR) followed by a line feed (LF).

Data Link A connection between any two fields in source and target databases capable of sending and receiving information, such as fields from the Source database linked to Target database fields.

Database Manager Part of the Administrator and Architect that allows you to add, view, and delete tables and fields in the database.

Defect A record describing a problem in a product or process, or a feature request for a product or process.

Function A set of instructions that operates as a single logical unit, can be called by name, accepts input parameters, and returns information. In programming languages such as C, a function is a named subroutine of a program that encapsulates some logic. The function can be called by name, using parameters to pass data in to the function and retrieve data produced by the function. For example, a component of a formula used by the Translation Workbench to change or update fields in a database.

Groups Groups help to manage your workflow. You can create groups to work with a subset of accounts, contacts, tickets, defects, etc. You can create a group based on a specific set of conditions, or choose individual accounts, tickets, contacts, etc. from the database. Groups can be created within the Architect.

Index In a relational database, a database object that provides fast access to data in the rows of a table, based on key values. Indexes provide quick access to data and can enforce uniqueness on the rows in a table.

Join As a verb, to combine the contents of two or more tables and produce a result set that incorporates rows and columns from each table. Tables are typically joined using data that they have in common. As a noun, the process or result of joining tables, as in the term "inner join" to indicate a particular method of joining tables. Joins are managed with the Global Join Manager in the Architect or the Administrator. You can join tables using data that exists in both tables or with fields having different names but the same content. Once your joins are created, you can query on any field in either table. These joins can be used when creating or modifying views and reports.

List View A list view is a hybrid of a detail view and a standard list view. The detail portion of the list view allows you to see important detail information for the record currently in the list portion of the view. Records opened from a list view launch a specific detail view for editing. List views can be managed from within the Architect.

Lookups Lookups enable you to search for information under any of the major families (for example; Account, Contact, Ticket, and so on). You can initiate a lookup to find information that shares certain characteristics. Once the lookup is created, you can save the result as a group. Sage SalesLogix users can access predefined lookups by clicking on certain field labels in the detail views of the client. Lookups can be managed from either the Architect or the Administrator.

Null A character code with a null value; literally, a character meaning "nothing." Although it is real in the sense of being recognizable, occupying space internally in the computer, and being sent or received as a character, a NULL character displays nothing, takes no space on the screen or on paper, and causes no specific action when sent to a printer. A field that does not contain data.

ODBC Acronym for open database connectivity which is an interface providing a common language for Windows applications to gain access to a database on a network. This is a DBMS API (DataBase Management System Application Programming Interface), a set of rules that allow programs to interact with a wide range of databases.

Parameter A placeholder in a query or stored procedure that can be filled when the query or stored procedure is executed. Parameters allow you to use the same query or stored procedure many times, each time with different values. Parameters can be used for any literal value, and in some databases, for column references as well.

Pick List A pick list is a list of values that you can select to enter data in a field. Pick lists encourage consistent data entry. A set of rules governs who can create and maintain pick lists. Pick lists can be created in the Architect and the Administrator.

Plugins A small software program that plugs into a larger application to provide added functionality. For example, components that customize and add functionality to Sage SalesLogix. Plugins include views, reports, templates, Basic scripts, and SQL scripts.

Primary Key The column or columns that are used to uniquely identify each row in a table. All values for a primary key are unique and non-null. Used when one table must refer to values in another table.

Procedure In a program, a named sequence of statements, often with associated constants, data types, and variables, that usually performs a single task. A procedure can usually be called (executed) by other procedures, as well as by the main body of the program. Some languages distinguish between a procedure and a function, with the latter returning a value. A set of instructions that describes how to handle a specific problem or answer a specific question.

Reports Sage SalesLogix uses Crystal Reports and its associated runtime files for creating and viewing reports. Reports can be based on any standard table (for example; Accounts, Contacts, Tickets, Defects) or any custom table created in the DB Manager. See the Compatibility Checklist for information on the versions of Crystal Reports supported.

Required Field A field in a new record that must be filled before the record is saved. If a required field is blank, you cannot save changes to the record.

Return An agreement to accept returned products from a customer for replacement, refund, or credit.

Rollback A method of returning a transaction on a database to a prior state. All changes made to the object subsequent to the initiation of the transaction are voided, and the object remains in the state it was at the time of the beginning of the transaction.

Sage SalesLogix Client The core customer relationship management component of Sage SalesLogix. It is used to connect to the Sage SalesLogix database and enables users to access and manage tickets, contacts, accounts, opportunities, defects, contracts, and returns.

Security Profile Determines each user's access to information and functionality within Sage SalesLogix.

Site Code A unique identification code, assigned to each user and database, that represents the user or component.

SLXLogs A root directory of other folders (Documents and Library).

SpeedSearch Service SpeedSearch enables users to search for information in existing indexes.

Step An operation against a target database, which affects a single data object.

Stored Procedure A set of one or more SQL statements that are stored together in a database. Stored procedures can range from very simple to very complex.

Syntax The grammar of a language; the rules governing the structure and content of statements.

Team A group of users who have access to the same accounts. Members of the same team may have different read/write access to data.

Ticket A record of a call relating to a question or problem experienced by the customer.

Transaction A logically related sequence of SQL commands that accomplishes a particular result for an application. A transaction begins when the application starts or when a commit or rollback is executed. The transaction ends when the next commit or rollback is executed.

Truncate To cut off or eliminate all data that comes after the decimal point. Thus, if you truncate 1.2345, you get the value 1. If you truncate the value 1.9999 you also get the value 1. Truncating does not round data, it simply cuts off unwanted data.

Unique Index An index in which no two rows are permitted to have the same index value, thus prohibiting duplicate index or key values. The system checks for duplicate key values when the index is created and checks each time data is added with an INSERT or UPDATE statement.

Unique Key One or more columns that must be unique for each row of the table. An index that ensures that no identical key values are stored in the table.

View All views in the Sage SalesLogix clients can be modified to better fit your specific business model. As these views are essentially the "windows" to your data, you may want to include additional fields on the standard views, or create buttons that launch specific applications. These examples and more are possible within the Architect. Views are made up of objects, such as combo boxes and edit boxes. Using the Architect, you can modify the default views included with Sage SalesLogix, or create your own.

Index