ACT!™

Developer's Reference

ACT! Developer's Reference

Copyright Notice

Released: 8/2002 for ACT! version 6.0 for Windows.

Trademarks

Printed in the United States of America.

10 9 8 7 6 5 4 3 2 1

INTERACT COMMERCE CORPORATION LICENSE AND WARRANTY

NOTICE: INTERACT COMMERCE CORPORATION, A DIVISION OF BEST SOFTWARE ("INTERACT COMMERCE") LICENSES THE SOFTWARE TO YOU ONLY UPON THE CONDITION THAT YOU ACCEPT ALL OF THE TERMS CONTAINED IN THIS LICENSE AGREEMENT. PLEASE READ THE TERMS CAREFULLY BEFORE CONTINUING. IF YOU DO NOT AGREE TO THESE TERMS, INTERACT COMMERCE IS UNWILLING TO LICENSE THE SOFTWARE TO YOU.

USE OF SOFTWARE

The software that accompanies this license (the "Software") is the property of Interact Commerce Corporation, a division of Best Software ("Interact Commerce") or its licensors and is protected by copyright law and international treaty. While Interact Commerce or its licensors continues to own the Software, you will have certain rights to use the Software after your acceptance of this license. Except as may be modified by a written license addendum which accompanies this license, your rights and obligations with respect to the use of this Software are as follows:

You may:

(i) use the software for the purposes intended;

(ii) make one copy of the Software for archival purposes, or copy the software onto the hard disk of your computer and retain the original for archival purposes;

You may not:

(i) sell copies of the Software or the documentation, if any, that accompanies the Software;

(ii) rent or lease any portion of the Software or host the Software on your computer for others to use for a fee; or

(iii) reverse engineer, decompile, disassemble, modify, translate, make any attempt to discover the source code of the Software, or create derivative works from the Software;

NO WARRANTY

The Software is accepted by you "AS IS" AND "WITH ALL FAULTS." ALL WARRANTIES CONCERNING THE SOFTWARE, EXPRESS OR IMPLIED, STATUTORY, OR IN ANY OTHER PROVISION OF THIS AGREEMENT INCLUDING, WITHOUT LIMITATION, ANY WARRANTY OF TITLE, NON-INFRINGEMENT, MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE, ARE HEREBY EXPRESSLY DISCLAIMED AND EXCLUDED. YOUR SOLE AND EXCLUSIVE REMEDY FOR A BREACH OF THIS AGREEMENT BY INTERACT SHALL BE TO TERMINATE THIS AGREEMENT.

U.S. GOVERNMENT RESTRICTED RIGHTS

The Software is commercial in nature. The Software is a "Commercial Item", as that term is defined in 48 C.F.R. §252.227-7014(a)(5) and 48 C.F.R. §252.227-7014(a)(1), and used in 48 C.F.R. §12.212 and 48 C.F.R. §227.7202, as applicable. Consistent with 48 C.F.R. §12.212, 48 C.F.R. §252.27-7015, 48 C.F.R. §227.7202 through 227.7202-4, 48 C.F.R. §52.227-14, and other relevant sections of the Code of Federal Regulations, as applicable.

GENERAL

This Agreement will be governed by the laws of the State of Arizona, U.S.A. The U.N. Convention on the International Sale of Goods is expressly excluded. This Agreement may only be modified by a written document which has been signed by both you and Interact Commerce. Should you have any questions concerning this Agreement, or if you desire to contact Interact Commerce for any reason, please write: Interact Commerce Customer Sales and Service, 8800 N. Gainey Center Drive #200, Scottsdale, Arizona, 85258, USA.

License and Warranty Addendum

The software you are acquiring has been specially configured to run on a network, and consequently Interact Commerce Corporation grants to you the right to use the enclosed Software on a computer network provided that for each and every concurrent user on the network, you have acquired and dedicated one licensed copy of the software.

Contents

Introduction

The ACT! Software Development Kit (SDK) includes documentation and sample code files for SDK components that are built into the ACT! application. The ACT! SDK extends the functionality of ACT!, enables external applications to control ACT!, reads and writes to ACT! database tables, and adds auxiliary commands to the user interface to execute external programs. The ACT! SDK was developed for use by ACT! Certified Consultants, ACT! add-on product developers, independent software developers, and corporate developers.

About this manual

The *ACT! Developer's Reference* is written for developers, integrators, ACT! Certified Consultants, and information system (IS) professionals. Readers are assumed to be familiar with standard programming practices and tools.

The following table describes the documents included in the ACT! Developer's Reference.

Section	Document	Description
Introduction	Introduction	Introduction to the manual. Includes an overview, document conventions, and technical support information.
1	ACT! Databases	Details the file formats of ACT! database tables, relationships between tables, and contact and group import order.
2	The Database Class	Describes the Database class used to read and write information in ACT! database tables, and lists special considerations, properties, and methods common to all objects in the class.
3	Objects Derived from the Database Class	Defines and describes objects within the Database class, including methods, properties, and code samples.
4	The Application Class	Describes the Application class which provides non-ACT! applications with both control and context integration of the ACT! application. Also lists special considerations, properties, and methods common to all objects in the class.
5	Objects Derived from the Application Class	Defines and describes objects within the Application class, including methods, properties, and code samples.
6	The Scripting and Command Objects	Describes the ACT! OLE Scripting and Command objects. The Scripting object registers scripts for use with ACT!, and also includes event notification for use with scripting. The Command objects permits external applications to interact with ACT!.
7	Views and Tabs	Discusses how to add views, accessible by a Contact or Group tab or View command, to display HTML content within the ACT! application.
Appendix A	Folder Structure	Lists and describes all common ACT! folders and contents, including templates.
Appendix B	Command IDs	Lists ACT! command IDs referenced in the ACT! Application object and Command object sections of the ACT! SDK.

The following sample files are included with the ACT! Software Development Kit (SDK).

- Sample Microsoft Visual Basic and Visual C++ code files for the ACT! OLE Application object and the ACT! OLE Database object.
- Sample control files for adding extensible views and tabs to the ACT! application.
- A basic database schema HTML page which details the fields in primary ACT! tables and table relationships.

Document conventions

To clearly communicate SDK functionality, this document uses:

- Typographic conventions
- Standard definitions
- Common parameter types

Typographic conventions

The SDK documentation uses the following typographic conventions.

Convention	Description
Bold	Indicates command names, function names, properties, methods, data types, or other keywords in the programming interface or programming language. Type exactly as shown.
UPPERCASE	Indicates a flag, return value, property, file names, registers, and terms used at the operating-system command level.
`Courier`	Example code is shown in a monospaced Courier font. Comments in the code are preceded by an apostrophe ('Comment). If a line of code does not fit on a single line of the page, the remaining code is indented on the next line. Code examples are not case sensitive.
Italic	Parameters, return variables, data structure names and text that represent the type of text to be entered, rather than a literal series of characters, are shown in italic.
[Brackets]	Optional items in syntax statements are enclosed in brackets ([ ]). For example, [password] indicates that a password can be used with the command, but is not required. In commands, type only the information within the brackets, not the brackets.
Parameter1|Parameter2	Parameters are separated by a vertical bar (|) to indicate a mandatory choice between two items. Only one of the items can be specified.
Ellipsis...	Items that can repeat are indicated by an ellipsis (…) For example, devicename [...] indicates that optionally more than one device can be specified, separating device names with a space.
C:\Program Files\ACT\	Paths typically use the convention of ACT as the folder name that contains ACT! elements. In older installations, another folder, Symantec, immediately precedes the ACT folder, changing the path to: C:\Program Files\Symantec\ACT.

Standard terminology

ACT! database The term database, in relation to an ACT! database, refers to the set of tables and their associated indexes in an ACT! database.

Table The term table refers to a specific table in an ACT! database.

Common parameter types

The following data types are referenced throughout this document. Parameter names use Hungarian notation, beginning with a lowercase letter or letters that indicate the data type.

Parameter syntax	Data type
date	Date and time in Short Date style and Time style from Windows Regional Settings (DATE in Visual C++)
fParameter	Float (Single in Visual C++)
iParameter	Short integer
lParameter	Long integer
szString	String, terminated by a null character (BSTR in Visual C++)
TrueIFalse or *bParameter*	Boolean
vParameter	Variant (VARIANT in Visual C++)

Support

The ACT! Software Development Kit (SDK) offers two support options: "How-Do-I" support and Consulting services. For the latest information about support for the ACT! SDK, visit http://www.act.com/, click **Resources**, click **Add-On Products**, then click the **Software Development Kit**. Every effort has been made to ensure the accuracy of this information. Interact Commerce Corporation reserves the right to limit any single support call, change the terms and conditions of support, and change support pricing and service availability without notice.

To learn more about general ACT! Customer Service and Technical Support solutions, visit the Web sites listed in the *ACT! User's Guide* or choose Service and Support Information from the Help menu within ACT!. See the *ACT! User's Guide* for general Customer Service and Technical Support telephone numbers.

"How-Do-I" support

"How-Do-I" Technical Support for the ACT! SDK is charged at $50 US per incident. An incident is defined as a question regarding one specific operation. Please call (480) 444-1399 or (800) 927-3989 to leave a message for an ACT! SDK specialistYour call will be returned within two business days. "How-Do-I" support is available Monday through Friday, 6:00 A.M. to 5:00P.M. Pacific Standard Time.

Consulting services

Users of the ACT! SDK are also entitled to code debugging services. These services are charged at $100 US per hour. These services cover errors encountered during use of the ACT! SDK. Please call(480) 444-1399 or (800) 927-3989 to leave a message for an ACT! SDK specialist. Your call will be returned within two business days. Consulting services are available Monday through Friday, 6:00 A.M. to 5:00P.M. Pacific Standard Time.

Programming services are available from independent ACT! Certified Consultants. For a list of ACT! Certified Consultants, visit http://www.actsoftware.com/ then click the Certified Consultants link.

Chapter 1
ACT! Databases

This chapter describes the file formats of ACT! database tables and the relationships between the tables in ACT! 3.0 or later. It provides technical information on tables and the fields within to assist in developing applications that work with ACT! databases. Interact Commerce Corporation recommends using the ACT! Database class to read and write information to ACT! databases.

Overview

The ACT! database is a collection of files that store information entered by the user. This collection of files consists of tables that store contact, group, activity, notes and history, e-mail, and other information. The Relational table resolves many to many relationships, storing the links that allow this information to be accessible in associated tables. ACT! 3.0 or later uses standard FoxPro formatted tables.

The following table describes ACT! database files based on their extensions.

Extension	Description
.ADB	Activity table
.ADX	Index file for the Activity (.ADB) table
.BLB	Binary Large Object Database (BLOB) file
.DBF	Contact table
.DDB	List table for the Sales (.SDB) table (in ACT! 5.0 or later databases)
.DDF	Database definition file
.DDX	Index file for the List (.DDB) table (in ACT! 5.0 or later databases)
.EDB	E-mail table
.EDX	Index file for the E-mail (.EDB) table
.GDB	Group table
.GDX	Index file for the Group (.GDB) table
.HDB	Notes/History table
.HDX	Index file for the Notes/History (.HDB) table
.LCK	Record locking file for ACT! multi-user databases
.MDX	Index file for the Contact (.DBF) table
.REL	Relational table for ACT! tables
.REM	Reminders file (in ACT! 5.0 or later databases)
.REX	Index file for the Relational (.REL) table
.SDB	Sales table (in ACT! 5.0 or later databases)
.SDX	Index file for the Sales (.SDB) table (in ACT! 5.0 or later databases)

Extension	Description
.TDB	Transaction synchronization log table
.TDX	Index file for the Transaction synchronization log (.TDB) table
.TLX	Record locking file for ACT! multi-user databases on a network, where x is the number of each subsequent file (TL1, TL2, and so on)

The following list provides brief descriptions of ACT! database tables and an overview of data relationships within ACT! tables.

Activity (.ADB) Contains records for call, meeting, and to-do activities. Activity records are identified by a Unique ID and an activity type. Activity records are linked to contact records by Relational table records, and to group records by a Unique ID.

Binary Large Object Database (.BLB) Contains variable size data including attachment file and path names, database user information, drop-down list information, e-mail addresses, notes and history regarding text, recurring activity information, and synchronization settings. ACT! database tables contain references as links to a Binary Large Object Database (BLOB) file.

Contact (.DBF) Contains contact information. Contact records are identified by a Unique ID and linked with an ACT! database user by the User Name.

Database Definition File (.DDF) Stores field definition information including mapping of ACT! field names to ACT! database schema names, index file structures, and references to drop-down list items in the BLOB file.

E-mail (.EDB) Contains the e-mail address and e-mail system information that is displayed in the Contact window. E-mail records are identified and linked to contact records by the Contact Unique ID field. E-mail addresses are stored in the Binary Large Object Database file.

Group (.GDB) In ACT! 3.0 or later, contains separate records each with their own histories, notes, and attachments. Group records are identified by a Unique ID and linked with an ACT! database user by the User Name. Group records are linked to contact records by records in the Relational table.

List (.DDB) In ACT! 5.0 or later, contains product, product type, and main competitor data linked to the Sales table. The Sales table stores a Unique ID for the field containing each of these three types of data. List records are identified and linked to sales records by the List table Unique ID field.

Notes/History (.HDB) Contains notes, histories, and attachments. Note, history, and attachment records are identified by a Unique ID. Notes/History records and attachments are linked to contact and group records by Unique ID fields in Notes/History table records.

Record locking file (.LCK) The record locking file for ACT! multi-user databases.

Relational table (.REL) The Relational table links contact, group, and activity table records in ACT! 3.0 or later.

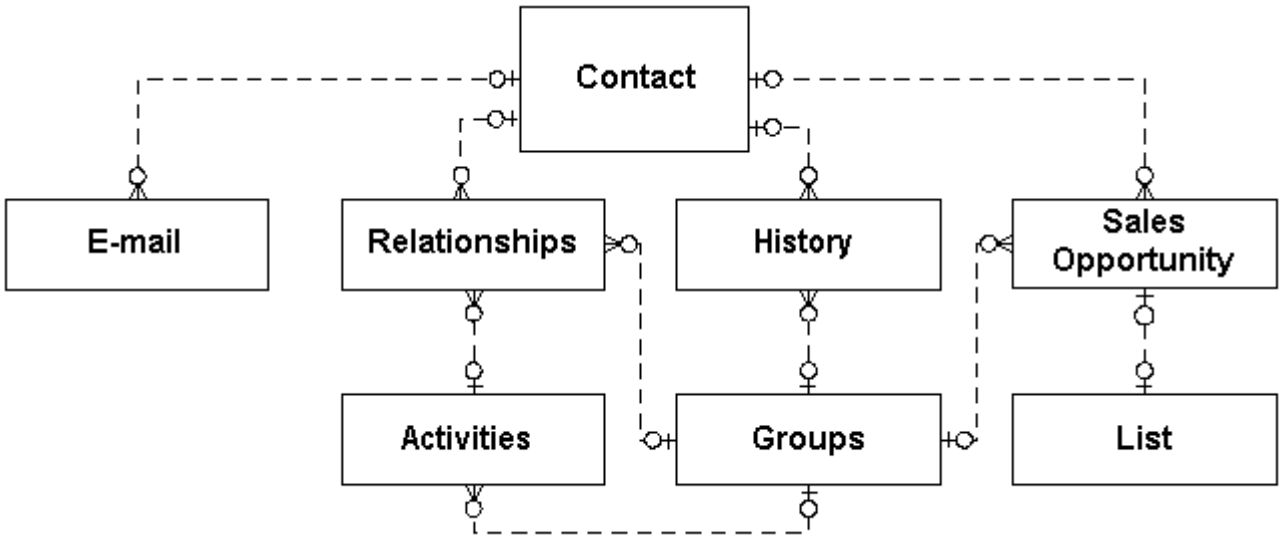
Reminders file (.REM) Contains ACT! reminders data supplied by the system.

Sales (.SDB) In ACT! 5.0 or later, contains sales information. Sales records are identified by a Unique ID. Sales records are linked to contact and group records and data in the List table by Unique ID fields in Sales table records.

Transaction Synchronization Log (.TDB) Records the fields that are changed in an ACT! database and the date and time of the change for synchronization with another database.

Database table relationships

The following diagram shows each table in an ACT! database and how these tables are linked, including links to the Relational table.



For more detailed information on table relationships, see the ACT! schema included with the SDK (ACT_Schema.htm).

ACT! database table fields also link to data stored in the Binary Large Object Database (BLOB) file. When BLOB data is required, SDK commands automatically retrieves it.

Looking at database tables

The tables in this section show database schema information for each field in every ACT! database table. This includes field names, database schema names, field formats, and field lengths. The tables also show the field ID values and the corresponding field constants to use to read and write data to ACT! database fields using the Database and Application classes.

The Unique ID field contains a 12 character value that is derived from the current date and time, then translated into a unique sequence of alphanumeric and special characters that can be displayed. User-defined fields other than the defaults start numbering at Unique ID type 1000. Also, any field whose Unique ID type is between 200 and 999 is a virtual field that is read-only. Virtual fields reference a calculated value, a portion of a field in the same table, or a field in another table or file.

Note To find the field IDs assigned to custom fields added to the database, use the ACTDIAG tool provided in the support folder.

Activity table (.ADB) fields

The following table shows database schema information for each field in an ACT! database Activity table.

Field name	Database schema name	Format/ Length	FieldID	Field constant	Description
Alarm Status	ALRMSTATUS	Num1	33	AF_AlarmStatus	Alarm status for the activity. Values are: 0 Alarm is set to off 1 Alarm is set to on
Banner Color	BANNER_CLR	Num10	34	AF_BannerColor	Code for the activity color. Default values are: Black - Low-priority Blue - Medium-priority Red - High-priority
Cleared Status	CLEARED	Num1	41	AF_ClearedStatus	Specifies if the activity was cleared. Values are: Blank - Not cleared 1 - Cleared
Contact Count	CONT_CNT	Num6	100	AF_TotalInActivity	Total number of contacts with whom the activity is scheduled. This field is supplied by the system.
Create Timestamp	CTIME	Char6	2	AF_CreateTimestamp	Date and time the activity record was created. This field is supplied by the system and stored in a compressed format.
Date/Time ("Date" and "Time" display)	START_TIME	Char12	28	AF_StartTime	Start date and time of the activity. The format displays in the table as YYYYMMDDHHMM. To set this field, use a standard date variable. The default is the current system date and time.
Details	DETAILS	Char6	45	AF_Details	Contains a description of the details associated with the activity for ACT! 5.0 or later databases only. This 6-byte field is supplied by the system and contains a reference to a field in the Binary Large Object Database file that contains a maximum of 32,768 characters.
Duration	DURATION	Num10	30	AF_Duration	Duration of the activity in minutes.
Edit Timestamp	ETIME	Char6	3	AF_EditTimestamp	Date and time the activity record was last modified. The initial value is the date and time the activity record was created. This field is supplied by the system and stored in a compressed format.
E-mail Status	EML_STATUS	Num1	35	AF_EmailStatus	Specifies if an e-mail reminder will be sent for the activity. Values are: 0 No reminder will be sent 1 A reminder will be sent

Field name	Database schema name	Format/ Length	FieldID	Field constant	Description
End Time	END_TIME	Char12	29	AF_EndTime	Calculated end date and time for the activity. The format is YYYYMMDDHHMM. This field is supplied by the system.
Exception Date	EXCEPTDATE	Char12	44	AF_ExceptionDate	Original date of an exception instance of a recurring activity. The format is YYYYMMDDHHMM.
External Id	EXTERNID	Char48	47	AF_ExternalId	Contains the record ID of an activity record in an external database (Outlook) in ACT! 5.0 or later databases only.
Group	GROUPID	Char12	39	AF_GroupId	The Unique ID of the group record to which the activity record is associated. This field is supplied by the system.
Lead Time	LEAD_TIME	Num10	32	AF_LeadTime	Advance notice for the activity alarm in minutes. The default lead time for the activity is used if the alarm is not set.
Merge Timestamp	MTIME	Char6	4	AF_MergeTimestamp	Date and time the activity record was imported into ACT! or synchronized with another ACT! database. This field is supplied by the system and stored in a compressed format.
Priority	PRIORITY	Num1	26	AF_Priority	Priority of the activity. Values are: 0 High 1 Medium 2 Low
Public/Private	PUB_STATUS	Num1	5	AF_PublicPrivate	Specifies if the activity is public or private.Values are: 1 Public 2 Private
Record Status	RECSTATUS	Num2	46	AF_RecordStatus	Specifies if the record was imported from or exported to an external database (Outlook), in ACT! 5.0 or later databases only.
Recurring Exceptions	EXCEPTIONS	Char6	43	AF_Exceptions	A list of dates for exception instances of a recurring activity. This 6-byte field is supplied by the system and contains a reference to data stored in the Binary Large Object Database file.
Recurring Identifier	RECURID	Char12	42	AF_RecurringId	For recurring activities, contains the unique Recurring Identifier of the parent activity if this instance of the recurring activity has been changed. Otherwise this field is blank. This field is supplied by the system.
Recurring Map	RECURRING	Char18	36	AF_Recurring	Recurring activity settings of:once (default), daily, weekly, monthly, or custom.

Field name	Database schema name	Format/ Length	FieldID	Field constant	Description
Regarding	REGARDING	Char70	27	AF_Regarding	Description of the activity.
Scheduled By	SCHEDL_BY	Char12	38	AF_ScheduledBy	User ID of the database user who scheduled the activity. The default is the logged-on user. This field is supplied by the system.
Scheduled Date		Char8	200	AVF_ScheduledDate	Date portion of the date and time for which the activity is scheduled. The format is YYYYMMDD. This field references the Date/Time field and is read-only.
Scheduled For	SCHEDL_FOR	Char12	37	AF_ScheduledFor	User ID of the database user for whom the activity is scheduled. The default is the logged-on user. This field is supplied by the system.
Scheduled Time		Char4	201	AVF_ScheduledTime	Time portion of the date and time for which the activity is scheduled. The format is HHMM. This field references the Date/Time field and is read-only.
Scheduled With	SCHEDLWITH	Char12	40	AF_FirstScheduledWith	User ID of the contact displayed in the Calendar, Activities tab, and Task List for the activity. This field is supplied by the system.
Timeless Status	TM_STATUS	Num1	31	AF_TimelessStatus	Timeless status for the start time of the activity.Values are: 0 Not timeless 1 Timeless
Total Duration	TDURATION	Num6	101	AF_TotalDuration	Total number of minutes from the start time of the first instance of the activity to the ending time of the last instance of the activity, minus one minute.
Type	TYPE	Num2	25	AF_Type	Activity record type.Values are: 0 Call 1 Meeting 2 To-do
Unique Id	UNIQUE_ID	Char12	1	AF_UniqueID	Unique activity record identification number. This field is supplied by the system.

Contact table (.DBF) fields

The following table shows database schema and Contact window information for each field in an ACT! database Contact table.

Field name	Database schema name	Format/ Length	FieldID	Field constant	Description
2nd Contact	NAME2	Char50	72	CF_Name2	Second contact's name.
2nd Last Reach (not displayed unless added in the Layout Designer)	ALT1REACH	Char8	92	CF_Alt1Reach	Date of the last completed call to the second contact. The format is YYYYMMDD. This field is supplied by the system.

Field name	Database schema name	Format/ Length	FieldID	Field constant	Description
2nd Phone	PHONE2	Char42	74	CF_Phone2	Second contact's phone number.
2nd Phone Ext.("Ext." displays)	PHONE2_EXT	Char8	83	CF_Phone2Ext	Extension for the second contact's phone number.
2nd Title	TITLE2	Char50	73	CF_Title2	Second contact's title.
3rd Contact	NAME3	Char50	75	CF_Name3	Third contact's name.
3rd Last Reach (not displayed unless added in the Layout Designer)	ALT2REACH	Char8	93	CF_Alt2Reach	Date of the last completed call to the third contact. The format is YYYYMMDD. This field is supplied by the system.
3rd Phone	PHONE3	Char42	77	CF_Phone3	Third contact's phone number.
3rd Phone Ext. ("Ext." displays)	PHONE3_EXT	Char8	84	CF_Phone3Ext	Extension for the third contact's phone number.
3rd Title	TITLE3	Char50	76	CF_Title3	Third contact's title.
Address 1 ("Address" displays)	ADDR1	Char50	27	CF_Address1	First line of the contact's primary address.
Address 2 (label is not displayed)	ADDR2	Char30	28	CF_Address2	Second line of the contact's primary address.
Address 3 (label is not displayed)	ADDR3	Char30	29	CF_Address3	Third line of the contact's primary address.
Alt Phone	ALTPHONE	Char42	71	CF_AltPhone	Contact's alternate phone number.
Alt Phone Ext. ("Ext." displays)	ALTEXT	Char8	82	CF_AltPhoneExt	Extension for the contact's alternate phone number.
Assistant	ASSISTANT	Char50	47	CF_Assistant	Name of the contact's assistant.
Asst. Phone	ASST_PHONE	Char42	86	CF_AsstPhone	Phone number of the contact's assistant.
Asst. Phone Ext. ("Ext." displays)	ASST_EXT	Char8	87	CF_AsstExt	Extension for the phone number of the contact's assistant.
Asst. Title	ASST_TITLE	Char50	85	CF_AsstTitle	Title of the contact's assistant.
City	CITY	Char30	30	CF_City	City in the contact's address.
Company	COMPANY	Char50	25	CF_Company	Contact's company name.
Contact	NAME	Char50	26	CF_Name	Contact's name.
Contact Type	CONT_TYPE	Num1	125	CF_ContactType	Contact record type.Values are: blank, 0, or 1 for normal 2 for "My Record"
Country	COUNTRY	Char25	33	CF_Country	Country in the contact's address.
Create Timestamp ("Create Date" displays)	CTIME	Char6	2	CF_CreateTimestamp	Date and time the contact record was created. This field is supplied by the system and stored in a compressed format.
Department	DEPARTMENT	Char50	88	CF_Department	Contact's department.
Edit Timestamp ("Edit Date" displays)	ETIME	Char6	3	CF_EditTimestamp	Date and time the contact record was last modified. This field is supplied by the system and stored in a compressed format.

Field name	Database schema name	Format/ Length	FieldID	Field constant	Description
Email Address		Char-	200	CVF_EmailAddress	The primary e-mail address for the contact. This field references the Logon field in the E-mail table for ACT! 4.0 or later databases only and is read-only.
Email Carrier		Char128	203	CVF_EmailCarrier	The e-mail Carrier portion of the primary e-mail address for the contact. This field references the Carrier field in the E-mail table for ACT! 3.0 databases only and is read-only.
Email Logon		Char-	202	CVF_EmailLogon	The e-mail Logon portion of the primary e-mail address for the contact. This field references the Logon field in the E-mail table for ACT! 3.0 databases only and is read-only.
Fax	FAX	Char42	36	CF_Fax	Contact's fax number.
Fax Ext. (not displayed unless added in the Layout Designer)	FAX_EXT	Char8	81	CF_FaxExt	Extension for the contact's fax number.
First Name (not displayed unless added in the Layout Designer)	FNAME	Char50	78	CF_FirstName	Contact's first name. This field is parsed by the system from the contact.
Home Address 1	ALTADDR1	Char50	65	CF_AltAddress1	First line of the contact's home address.
Home Address 2	ALTADDR2	Char30	66	CF_AltAddress2	Second line of the contact's home address.
Home City	ALTCITY	Char30	67	CF_AltCity	City in the contact's home address.
Home Country	ALTCOUNTRY	Char25	70	CF_AltCountry	Country in the contact's home address.
Home Phone	HOME_PHONE	Char42	37	CF_HomePhone	Contact's home phone number.
Home State	ALTSTATE	Char20	68	CF_AltState	State in the contact's home address.
Home Zip	ALTZIP	Char10	69	CF_AltZip	ZIP code in the contact's home address.
ID/Status	IDSTATUS	Char25	34	CF_IDStatus	Category assigned to the contact.
Last Attempt	LAST_ATMPT	Char8	43	CF_LastAttempt	Date of the last attempt to call the contact. The format is YYYYMMDD. This field is supplied by the system.
Last Meeting	LAST_MEET	Char8	41	CF_LastMeet	Date of the last meeting with the contact. The format is YYYYMMDD. This field is supplied by the system.
Last Name (not displayed unless added in the Layout Designer)	LNAME	Char50	79	CF_LastName	Contact's last name. This field is parsed by the system from the contact name.

Field name	Database schema name	Format/ Length	FieldID	Field constant	Description
Last Reach	LAST_REACH	Char8	42	CF_LastReach	Date of the last completed call to the contact. The format is YYYYMMDD. This field is supplied by the system.
Last Results	LAST_RSLTS	Char75	48	CF_LastResults	Comments on the last results with the contact.
Letter Date	LTTR_DATE	Char8	44	CF_LetterDate	Date of the last letter sent to the contact. The format is YYYYMMDD. This field is supplied by the system.
Merge Timestamp ("Merge Date" displays)	MTIME	Char6	4	CF_MergeTimestamp	Date and time the contact record was imported into ACT! or synchronized with another ACT! database. This field is supplied by the system and stored in a compressed format.
Mobile Phone	MOBILPHONE	Char42	38	CF_MobilePhone	Contact's mobile phone number.
Note		Char-	201	CVF_Note	This field is obsolete and is supplied for backward-compatibility with ACT! 2.0 databases.
Owner (not displayed unless added in the Layout Designer)	OWNER	Char50	91	CF_UsersCompany	The company name of the database user who created the contact record.
Pager	PAGER	Char42	39	CF_Pager	Contact's pager number.
Phone	PHONE	Char42	35	CF_Phone	Contact's primary phone number.
Phone Ext. ("Ext." displays)	EXT	Char8	80	CF_Ext	Extension for the contact's primary phone number.
Public/Private	PUB_STATUS	Num1	5	CF_PublicPrivate	Access level for the contact.Values are: 1 Public (default) 2 Private
Record Creator	CREATOR	Char50	90	CF_Creator	The database user who created the contact record. This field is supplied by the system.
Record Manager	USER	Char12	6	CF_RecordManager	The Unique ID of the database user permitted to access and change the status of private contacts. This field is supplied by the system.
Referred By	REFER_BY	Char30	49	CF_ReferredBy	Description of the contact's referral source.
Salutation	SALUTATION	Char30	40	CF_Salutation	Contact's letter salutation or greeting name.
Spouse	SPOUSE	Char50	89	CF_Spouse	Name of the contact's spouse.
State	STATE	Char20	31	CF_State	State in the contact's address.
Ticker Symbol	TICKERSYM	Char12	95	CF_TickerSymbol	Company's stock ticker symbol for ACT! 4.0 or later databases only.
Title	TITLE	Char50	46	CF_Title	Contact's title.

Field name	Database schema name	Format/ Length	FieldID	Field constant	Description
Unique Id	UNIQUE_ID	Char12	1	CF_UniqueID	Unique contact record identification number. This field is supplied by the system.
User 1	USER1	Char50	50	CF_User1	User-definable field 1.
User 2	USER2	Char50	51	CF_User2	User-definable field 2.
User 3	USER3	Char50	52	CF_User3	User-definable field 3.
User 4	USER4	Char50	53	CF_User4	User-definable field 4.
User 5	USER5	Char50	54	CF_User5	User-definable field 5.
User 6	USER6	Char50	55	CF_User6	User-definable field 6.
User 7	USER7	Char75	56	CF_User7	User-definable field 7.
User 8	USER8	Char75	57	CF_User8	User-definable field 8.
User 9	USER9	Char75	58	CF_User9	User-definable field 9.
User 10	USER10	Char50	59	CF_User10	User-definable field 10.
User 11	USER11	Char50	60	CF_User11	User-definable field 11.
User 12	USER12	Char50	61	CF_User12	User-definable field 12.
User 13	USER13	Char50	62	CF_User13	User-definable field 13.
User 14	USER14	Char50	63	CF_User14	User-definable field 14.
User 15	USER15	Char50	64	CF_User15	User-definable field 15.
Web Site	URL	Char75	94	CF_URL	Contact's web site URL address.
Zip	ZIP	Char10	32	CF_Zip	ZIP code in the contact's address.
Birth date	BIRTHDAY	Char8	98	CF_BIRTHDAY	Stores the birthdate of a contact as an annual event. The format is: YYYYMMDD. Only available in new ACT! 6.0 databases.

E-mail table (.EDB) fields

The following table shows database schema information for each field in an ACT! database E-mail table. An e-mail record is created for each e-mail address for a contact.

Field name	Database schema name	Format/ Length	FieldID	Field constant	Description
Contact	CONTACTID	Char12	28	EF_ContactId	The Unique ID of the contact record with which the e-mail record is associated. This field is supplied by the system.
Create Timestamp	CTIME	Char6	2	EF_CreateTimestamp	Date and time the e-mail record was created. This field is supplied by the system and stored in a compressed format.
Edit Timestamp	ETIME	Char6	3	EF_EditTimestamp	Date and time the e-mail record was last modified. The initial value is the date and time the e-mail record was created. This field is supplied by the system and stored in a compressed format.

Field name	Database schema name	Format/ Length	FieldID	Field constant	Description
Logon	LOGON	Char6	25	EF_Logon	The e-mail address. This 6-byte field is supplied by the system and contains a reference to the Binary Large Object Database file.
Carrier	CARRIER	Char128	26	EF_Carrier (not used in ACT! 4.0 or later)	The e-mail system for the e-mail address in this record. This field is blank in ACT! 4.0 or later databases. In ACT! 4.0 or later, the e-mail system is a preference and is stored in the Windows registry.
Merge Timestamp	MTIME	Char6	4	EF_MergeTimestamp	Date and time the e-mail record was imported into ACT! or synchronized with another ACT! database. This field is supplied by the system and stored in a compressed format.
Primary Status	PRM_STATUS	Num1	27	EF_PrimaryStatus	E-mail record status.Values are: 0 Secondary e-mail address 1 Primary e-mail address
Unique Id	UNIQUE_ID	Char12	1	EF_UniqueID	Unique e-mail record identification number. This field is supplied by the system.

Group table (.GDB) fields

The following table shows database schema and Group window information for each field in an ACT! database Group table.

Field name	Database schema name	Format/ Length	FieldID	Field constant	Description
Address 1	ADDR1	Char50	27	GF_Address1	First line of the group's address.
Address 2	ADDR2	Char30	28	GF_Address2	Second line of the group's address.
Address 3	ADDR3	Char30	29	GF_Address3	Third line of the group's address.
City	CITY	Char30	30	GF_City	City in the group's address.
Contact Count	CONT_CNT	Num6	100	GF_TotalInGroup	Total number of contacts in the group.
Country	COUNTRY	Char25	33	GF_Country	Country in the group's address.
Create Timestamp ("Create Date" displays)	CTIME	Char6	2	GF_CreateTimestamp	Date and time the group record was created. This field is supplied by the system and stored in a compressed format.
Description	DESCRPTION	Char100	40	GF_Description	Description of the group.
Division	DIVISION	Char50	26	GF_Division	Division for the group.
Edit Timestamp ("Edit Date" displays)	ETIME	Char6	3	GF_EditTimestamp	Date and time the group record was last modified. This field is supplied by the system and stored in a compressed format.
Group Level	GRPLEVEL	Num1	56	GF_GroupLevel	The level of the group (a parent group or a subgroup) for ACT! 5.0 or later databases only.Values are: 0 Parent group 1 Subgroup.

Field name	Database schema name	Format/ Length	FieldID	Field constant	Description
Group Name	GRP_NAME	Char75	25	GF_Name	Name of the group.
Industry	INDUSTRY	Char50	58	GF_Industry	Type of industry for the group for ACT! 5.0 or later databases only.
Merge Timestamp ("Merge Date" displays)	MTIME	Char6	4	GF_MergeTimestamp	Date and time the group record was imported into ACT! or synchronized with another ACT! database. This field is supplied by the system and stored in a compressed format.
Note		Char-	200	GVF_Note	This field is obsolete and is supplied for backward-compatibility with ACT! 2.0 databases.
Number of Employees	EMPLOY	Num10	60	GF_Employees	Number of employees in the group for ACT! 5.0 or later databases only.
Parent ID	PARENTID	Char12	55	GF_ParentId	The Unique ID of the parent record of a subgroup record for ACT! 5.0 or later databases only. This field is supplied by the system.
Parent Name		Char75	201	GVF_ParentName	The group name of the parent record of a subgroup record for ACT! 5.0 or later databases only. This field is retrieved from the Parent ID field and is read-only.
Priority	PRIORITY	Char25	35	GF_Priority	User defined description of the group's priority. Default selections are High, Medium, and Low.
Public/Private	PUB_STATUS	Num1	5	GF_PublicPrivate	Access level for the group.Values are: 1 Public (default) 2 Private
Record Creator	CREATOR	Char50	54	GF_Creator	The database user who created the group record. This field is supplied by the system.
Record Manager	USER	Char12	6	GF_RecordManager	The database user permitted to access and change the status of private groups.
Referred By	REFER_BY	Char30	64	GF_ReferredBy	Description of the group's referral source for ACT! 5.0 or later databases only.
Region	REGION	Char50	57	GF_Region	Description of the geographic region of the group for ACT! 5.0 or later databases only.
Revenue	REVENUE	Curr18	61	GF_Revenue	Revenue for the group for ACT! 5.0 or later databases only, stored with 5 digits to the right of the decimal point.
SIC Code	SICCODE	Char20	59	GF_SicCode	SIC (Standard Industrial Classification) code for the group's industry for ACT! 5.0 or later databases only.
State	STATE	Char20	31	GF_State	State in the group's address.
Ticker Symbol	TICKERSYM	Char12	63	GF_TickerSymbol	Group's stock ticker symbol for ACT! 5.0 or later databases only.
Unique Id	UNIQUE_ID	Char12	1	GF_UniqueID	Unique group record identification number. This field is supplied by the system.

Field name	Database schema name	Format/ Length	FieldID	Field constant	Description
User 1	USER1	Char50	36	GF_User1	User-definable field 1.
User 2	USER2	Char50	37	GF_User2	User-definable field 2.
User 3	USER3	Char50	38	GF_User3	User-definable field 3.
User 4	USER4	Char50	39	GF_User4	User-definable field 4.
User 5	USER5	Char50	47	GF_User5	User-definable field 5.
User 6	USER6	Char75	48	GF_User6	User-definable field 6.
Web Site	URL	Char75	65	GF_URL	Group's web site URL address for ACT! 5.0 or later databases only.
Zip	ZIP	Char10	32	GF_Zip	ZIP code in the group's address.

List table (.DDB) fields

The following table shows database schema for each field in an ACT! database List table. (Requires ACT! 5.0 or later.)

Field name	Database schema name	Format/ Length	FieldID	Fieldconstant	Description
Create Timestamp	CTIME	Char6	2	LTF_CreateTime	Date and time the list record was created. This field is supplied by the system and stored in a compressed format.
Directory Name	DIRNAME	Char100	26	LTF_Name	Stores a drop-down list entry for a Product, Type or Main Competitor field referenced by a Unique ID in the Sales table.
Directory Type	DIRTYPE	Num2	25	LTF_Type	Associates the Directory Name with one of three specific fields referenced by a Unique ID in the Sales table.Values are: 1 Product 2 Type 3 Main Competitor
Edit Timestamp (System field)	ETIME	Char6	3	LTF_EditTime	Date and time the list record was last modified. The initial value is the date and time the record was created. This field is supplied by the system and stored in a compressed format.
Merge Timestamp (System field)	MTIME	Char6	4	LTF_MergeTime	Date and time the list record was imported into ACT! or synchronized with another ACT! database. This field is supplied by the system and stored in a compressed format.
Record Manager	USER	Char12	6	LTF_UserId	Unique ID of the database user who created the list record. This field is supplied by the system.
Unique Id (System field)	UNIQUE_ID	Char12	1	LTF_UniqueID	Unique list record identification number. This field is supplied by the system.

Notes/History table (.HDB) fields

The following table shows database schema information for each field in an ACT! database Notes/History table.

Field name	Database schema name	Format/ Length	FieldID	Fieldconstant	Description
Attachment	ATTACHMENT	Char6	28	NHF_Attachment	The drive, folder, and filename of the attached file. This 6-byte field is supplied by the system and contains a reference to a field in the Binary Large Object Database file that contains a maximum of 256 characters.
Contact	CONTACTID	Char12	29	NHF_ContactId	The Unique ID of the contact record with which the history, notes, or attachment record is associated. This field is supplied by the system.
Create Timestamp	CTIME	Char6	2	NHF_CreateTimestamp	Date and time the history, notes, or attachment record was created. This field is supplied by the system and stored in a compressed format.
Edit Timestamp	ETIME	Char6	3	NHF_EditTimestamp	Date and time the history, notes, or attachment record was last modified. The initial value is the date and time the record was created. This field is supplied by the system and stored in a compressed format.
Group	GROUPID	Char12	30	NHF_GroupId	The Unique ID of any group record with which the history, notes, or attachment record is associated. This field is supplied by the system.
Merge Timestamp	MTIME	Char6	4	NHF_MergeTimestamp	Date and time the history, notes, or attachment record was imported into ACT! or synchronized with another ACT! database. This field is supplied by the system and stored in a compressed format.
Recorded Date		Char8	200	NHVF_RecordedDate	The date portion of the User Time field. The format is YYYYMMDD. This field is retrieved from the User Time field and is read-only.
Recorded Time		Char4	201	NHVF_RecordedTime	The time portion of the User Time field. The format is HHMM. This field is retrieved from the User Time field and is read-only.
Regarding	REGARDING	Char6	26	NHF_Text	Description of the history event or the attachment, or the text of the note. This 6-byte field is supplied by the system and contains a reference to a field in the Binary Large Object Database file that contains a maximum of 30,000 characters. The next table shows descriptions of the regarding text by type number.

Field name	Database schema name	Format/ Length	FieldID	Fieldconstant	Description
Type	TYPE	Num3	25	NHF_Type	History event, note, or attachment type number. The next table shows values for the type numbers.
Unique Id	UNIQUE_ID	Char12	1	NHF_UniqueID	Unique notes, history, or attachment record identification number. This field is supplied by the system.
Record Manager	USER	Char12	6	NHF_UserId	The Unique ID of the database user permitted to access and change the status of private activities. This field is supplied by the system.
User Time (Date and Time display)	USER_TIME	Char12	27	NHF_UserTime	Date and time the notes, history, or attachment record was created. The format is YYYYMMDDHHMM. A different date and time can be specified when an activity is cleared to history.

The following table shows type values and regarding text for the Regarding and Type fields in the Notes/History table.

Type	Regarding text	Type	Regarding text
0	Call Attempted	15	Delete Activity
1	Call Completed	16	E-mail Sent
2	Call Received	17	Call Left Message
3	Field Changed	50	Fax Sent
4	Access	51	Sent Sync
5	Letter Sent	52	Received Sync
6	Meeting Held	53	Replace Fields Log
7	Meeting Not Held	54	To-do Erased
8	To-do Done	55	Meeting Erased
9	To-do Not Done	56	Error
10	Timer	57	Closed/Won Sale (for ACT! 5.0 or later)
11	Call Erased	58	Lost Sale (for ACT! 5.0 or later)
12	Contact Deleted	100	Note
13	Update Contact	101	Attachment
14	Update Activity	102	E-mail (for ACT! 4.0 or later)

Relational table (.REL) fields

The following table shows database schema for each field in an ACT! database Relational table, which resolves many to many relationships. All fields in this table are supplied by the system.

Field name	Database schema name	Format/ Length	Description
Create Time Stamp	CTIME	Char6	Date and time the relational table database record was created. This field is currently blank and reserved for future use.
Edit Time Stamp	ETIME	Char6	Date and time the relational table database record was last modified. This field is currently blank and reserved for future use.
Field1 (ID)	FIELD1	Char12	For type 0 and 1 records, contains the Unique ID of the record in the contact or group database associated or linked with the group or activity record Unique ID in Field2(ID).For type 3 records, contains the Unique ID of the database user associated with the activity record Unique ID in Field2(ID) of the activity, which has an alarm for the user for whom the activity is scheduled.
Field2 (ID)	FIELD2	Char12	Unique ID of the record in the group or activity database that is associated with the contact or group record Unique ID in Field1(ID).
Field3 (Time)	TIME	Char12	The activity alarm time. The format is YYYYMMDDHHMM.
Field4 (Time)	TIME2	Char12	The activity scheduled date. The format is YYYYMMDDHHMM. This field is blank if an alarm is not set for the activity.
Type	TYPE	Char1	Database relationship type number.Values are: 0 Links contact and group database records 1 Links contact and activity database records 3 Links contact database records with alarms

Understanding the Relational table

The following table shows relationship type values and descriptions in the Relational table.

Type	Field1 (ID)	Field2 (ID)	Description
0	Contact Unique ID	Group Unique ID	Links contact and group database records
1	Contact Unique ID	Activity Unique ID	Links contact and activity database records
3	Contact Unique ID of the database user for whom the activity is scheduled	Activity Unique ID	Links contact database records with alarms

Sales table (.SDB) fields

The following table shows database schema and Sales tab information for each field in an ACT! database Sales table. The Sales table exists only in ACT! 5.0 or later.

Field name	Database schema name	Format/ Length	FieldID	Fieldconstant	Description
Amount	SAMOUNT	Num19	34	SLF_Amount	The total amount for the sale, stored with 5 digits to the right of the decimal point.
Competitors		Char100	202	SLVF_Competitors	Name of the main competitor for the sale. This field is retrieved from the List table and is read-only.

Field name	Database schema name	Format/ Length	FieldID	Fieldconstant	Description
Contact	CONTACTID	Char12	25	SLF_ContactId	The Unique ID of the contact record with which the sales record is associated. This field is supplied by the system.
Create Timestamp	CTIME	Char6	2	SLF_CreateTime	Date and time the record was created. This field is supplied by the system and stored in a compressed format.
Creation Date	STARTDATE	Char8	35	SLF_SaleStartDate	Date on which the sales forecast was created. The format is YYYYMMDD.
Details	NOTES	Char6	36	SLF_Notes	Description of the sale or sales opportunity. This 6-byte field is supplied by the system and contains a reference to a field in the Binary Large Object (BLOB) Database file that contains a maximum of 32,768 characters.
Edit Timestamp	ETIME	Char6	3	SLF_EditTime	Date and time the record was last modified. The initial value is the date and time the record was created. This field is supplied by the system and stored in a compressed format.
Forecasted close date	SDATE	Char8	31	SLF_SaleDate	Forecasted or actual close date for the sale. The format is YYYYMMDD.
Group	GROUPID	Char12	26	SLF_GroupId	The Unique ID of the group record with which the sales record is associated. This field is supplied by the system.
Main Competitor	COMPETID	Char12	37	SLF_CompetitorsId	The Unique ID of the main competitor for the sale. This field is supplied by the system and references a competitor stored in the List table.
Merge Timestamp	MTIME	Char6	4	SLF_MergeTime	Date and time the record was imported into ACT! or synchronized with another ACT! database. This field is supplied by the system and stored in a compressed format.
Price	SPRICE	Num19	33	SLF_UnitPrice	The price per unit, stored with 5 digits to the right of the decimal point.
Probability	PROBABIL	Num4	30	SLF_Probability	Probability of making the sale as a percentage from 0 to 100.
Product Id	PRODID	Char12	28	SLF_ProductId	The Unique ID for the name of the product. This field is supplied by the system.
ProductName		Char100	200	SLVF_ProductName	Name of the product for the sale. This field is retrieved from the List table and is read-only.
Reason	REASON	Char65	38	SLF_Reason	Reason that the sale was closed/won or lost.
Record Manager	USER	Char12	6	SLF_UserId	The Unique ID of the Record Manager for the sale. This field is supplied by the system.

Field name	Database schema name	Format/ Length	FieldID	Fieldconstant	Description
Sales Stage	SALESSTAGE	Char40	39	SLF_SalesStage	Stage of the sale in the sales process (New Opportunity, Pre-Approach, and so on).
Status	SLSTATUS	Num1	27	SLF_Status	Status of the sale.Values are: 0 Sales opportunity 1 Closed/Won Sale 2 Lost Sale
Type	SLTYPE	Char12	29	SLF_TypeId	The Unique ID of the type for the sale. This field is supplied by the system and references a type stored in the List table.
TypeName		Char100	201	SLVF_TypeName	Name of the product type for the sale. This field is retrieved from the List table and is read-only.
Unique Id	UNIQUE_ID	Char12	1	SLF_UniqueID	Unique sales record identification number. This field is supplied by the system.
Units	SUNITS	Num14	32	SLF_Units	Number of units sold or expected to sell.

Importing and exporting ACT! data

This section describes how each field in a contact or group table is imported and exported. It describes field mapping for importing data and the default order of fields for exporting data. For procedures for importing and exporting data, see the *ACT! User's Guide* or type `importing` or `exporting` into the Index tab of the ACT! online help.

Contact table delimited text import field mapping

The following information is provided:

Field name. The default name of the field, as shown in an ACT! layout. Use Define Fields to change most field names.

Delimited text import mapping. Indicates that the field can be mapped when a delimited text file is imported into ACT! v5.0 or later databases.

Field name	Delimited text import mapping	Comments
2nd Contact	Yes	
2nd Last Reach	Yes	
2nd Phone	Yes	
2nd Phone Ext.	Yes	
2nd Title	Yes	
3rd Contact	Yes	
3rd Last Reach	Yes	
3rd Phone	Yes	
3rd Phone Ext.	Yes	

Field name	Delimited text import mapping	Comments
3rd Title	Yes	
Address 1	Yes	
Address 2	Yes	
Address 3	Yes	
Alt Phone	Yes	
Alt Phone Ext.	Yes	
Assistant	Yes	
Asst. Phone	Yes	
Asst. Phone Ext.	Yes	
Asst. Title	Yes	
City	Yes	
Company	Yes	
Contact	Yes	
Contact Type	No	System field defined by ACT!
Country	Yes	
Create Date	No	The current date is used.
Department	Yes	
Edit Date	No	The current date is used.
E-mail Login	Yes	E-mail Login maps to E-mail Address.
E-mail System (for fields in E-mail table and Binary large object database file)	Yes	
Fax	Yes	
Fax Ext.	Yes	
First Name	Yes	
Home Address 1	Yes	
Home Address 2	Yes	
Home City	Yes	
Home Country	Yes	
Home Phone	Yes	
Home State		
(County / Land / Province)	Yes	
Home Zip (Postcode)	Yes	
ID/Status	Yes	
Last Attempt	No	Field is blank.
Last Meeting	No	Field is blank.
Last Name	Yes	
Last Reach	No	Field is blank.
Last Results	Yes	

Field name	Delimited text import mapping	Comments
Letter Date	No	Field is blank.
Merge Date	No	The current date is used.
Mobile Phone	Yes	
Owner	Yes	
Pager	Yes	
Phone	Yes	
Phone Ext.	Yes	
Public/Private	No	The default of public is used.
Record Creator	No	The User Name of the logged-on user is used.
Record Manager	No	The User Name of the logged-on user is used.
Referred By	Yes	
Salutation	Yes	
Spouse	Yes	
State (County / Land / Province)	Yes	
Ticker Symbol	Yes	
Title	Yes	
Unique ID	No	System field defined by ACT!
User 1	Yes	
User 2	Yes	
User 3	Yes	
User 4	Yes	
User 5	Yes	
User 6	Yes	
User 7	Yes	
User 8	Yes	
User 9	Yes	
User 10	Yes	
User 11	Yes	
User 12	Yes	
User 13	Yes	
User 14	Yes	
User 15	Yes	
Web Site	Yes	
Zip (Postcode)	Yes	
Birth date	Yes	

Group table delimited text import field mapping

The following information is provided:

Field name. The default name of the field, as shown in an ACT! layout. Use Define Fields to change most field names.

Delimited text import mapping. Indicates that the field can be mapped when a delimited text file is imported into ACT! v5.0 or later databases.

Field name	Delimited text import mapping	Comments
Address 1	Yes	
Address 2	Yes	
Address 3	Yes	
City	Yes	
Country	Yes	
Contact Count	No	System field defined by ACT!
Create Date	No	The current date is used.
Description	Yes	
Division	Yes	
Edit Date	No	The current date is used.
Group Level	Yes	
Group Name	Yes	
Industry	Yes	
Merge Date	No	The current date is used.
Number of Employees	Yes	
Parent ID	Yes	
Priority	Yes	
Public/Private	No	The default of public is used.
Record Creator	No	The User Name of the logged-on user is used.
Record Manager	No	The User Name of the logged-on user is used.
Referred By	Yes	
Region	Yes	
Revenue	Yes	
SIC Code	Yes	
State (County / Land / Province)	Yes	
Ticker Symbol	Yes	
Unique ID	No	System field defined by ACT!
User 1	Yes	
User 2	Yes	
User 3	Yes	
User 4	Yes	
User 5	Yes	

Field name	Delimited text import mapping	Comments
User 6	Yes	
Web Site	Yes	
Zip (Postcode)	Yes	

Contact table default delimited text export field order

These ACT! Contact table fields can be exported to a delimited text file, in the default output file field order listed below. The only Contact table fields that cannot be exported are system fields Contact Type and Unique ID.

The following information is provided:

Field no. The default sequential number of the field as it appears in the delimited text output file.

Field name. The default name of the field, as shown in an ACT! layout. Use Define Fields to change most field names.

Field no.	Field name	Field no.	Field name
1	Public/Private	40	User 14
2	Record Manager	41	User 15
3	Company	42	Home Address 1
4	Contact	43	Home Address 2
5	Address 1	44	Home City
6	Address 2	45	Home State (County / Land / Province)
7	Address 3	46	Home Zip (Postal code)
8	City	47	Home Country
9	State (County / Land / Province)	48	Alt Phone
10	Zip (Postal code)	49	2nd Contact
11	Country	50	2nd Title
12	ID/Status	51	2nd Phone
13	Phone	52	3rd Contact
14	Fax	53	3rd Title
15	Home Phone	54	3rd Phone
16	Mobile Phone	55	First Name
17	Pager	56	Last Name
18	Salutation	57	Phone Ext.
19	Last Meeting	58	Fax Ext.
20	Last Reach	59	Alt Phone Ext.
21	Last Attempt	60	2nd Phone Ext.
22	Letter Date	61	3rd Phone Ext.
23	Title	62	Asst. Title
24	Assistant	63	Asst. Phone

Field no.	Field name	Field no.	Field name
25	Last Results	64	Asst. Phone Ext.
26	Referred By	65	Department
27	User 1	66	Spouse
28	User 2	67	Record Creator
29	User 3	68	Owner
30	User 4	69	2nd Last Reach
31	User 5	70	3rd Last Reach
32	User 6	71	Web Site
33	User 7	72	Ticker Symbol
34	User 8	73	Create Date
35	User 9	74	Edit Date
36	User 10	75	Merge Date
37	User 11	76	E-mail Login (from E-mail table)
38	User 12	77	E-mail System (from Binary large object database file)
39	User 13	78	Birth date

Group table default delimited text export field order

These ACT! Group table fields can be exported to a delimited text file, in the default output file field order listed below. The only Group table fields that cannot be exported are system fields Contact Count and Unique ID.

The following information is provided:

Field no. The default sequential number of the field as it appears in the delimited text output file.

Field name. The default name of the field, as shown in an ACT! layout. Use Define Fields in ACT! to change most field names.

Field no.	Field name	Field no.	Field name
1	Public/Private	18	User 5
2	Record Manager	19	User 6
3	Group Name	20	Record Creator
4	Division	21	Parent ID
5	Address 1	22	Group Level
6	Address 2	23	Region
7	Address 3	24	Industry
8	City	25	SIC Code
9	State (County / Land / Province)	26	Number of Employees
10	Zip (Postcode)	27	Revenue
11	Country	28	Ticker Symbol
12	Priority	29	Referred By

Field no.	Field name	Field no.	Field name
13	User 1	30	Web Site
14	User 2	31	Create Date
15	User 3	32	Edit Date
16	User 4	33	Merge Date
17	Description		

Chapter 2
The Database Class

This chapter describes the ACT! Database class component of the ACT! Software Development Kit (SDK), including common properties and methods used by objects within the class.

It is written for programmers to build applications that can interact with an ACT! database. To use this chapter, know the following:

- ACT! for Windows, version 3.0.3 or later
- Microsoft Windows 95/98/2000/Me/XP, or Windows NT 4.0.
- Microsoft Visual Basic, version 4.0 or later, or Microsoft Visual C++ 4.0 or later

Microsoft Visual Basic for Applications can be used as an alternative development language to access data in Microsoft Access and Excel versions in Microsoft Office 95, and Microsoft Access, Excel and Word versions in Microsoft Office 97 through Office 2002.

Overview

Developers can access and manipulate ACT! data using the Database class, also known as the ACT! OLE Database object. The primary function is to allow non-ACT! applications to open and access an ACT! database while maintaining the integrity of the database, including its indexes and tables. It also allows for simultaneous access to the databases by foreign applications while ACT! has the database open.

The Database class ensures that when an ACT! database is accessed to read or change data, all default values and rules will operate, unless the defaults are intentionally overwritten. Also, the Database class provides developers with a set of tools in the form of OLE automation objects with associated methods and properties. OLE facilitates application integration by defining a set of standard interfaces (groupings of semantically related functions) through which one application accesses the services of another.

Note Direct access to ACT! databases via other application drivers or direct manipulation performed inside foreign applications is not documented or supported and is strongly discouraged.

This object provides only for data access and has no links to the ACT! user interface in the OLE Database object. User interface access is provided by the ACT! Application class and layout editors internal to the program. In some cases, the ACT! UI can detect the update in its views if that database is open when foreign programs use the Database object to add data.

Note The Database class requires the proper version of ACT! be installed on the user's system.

Features and limitations

The Database object only works with 32-bit developer tools, such as Microsoft Visual Basic 4.0 or later, Microsoft Visual C++ 4.0 or later, or Microsoft Visual Basic for Applications.

The Database object provides the following functions:

- The ability to open and close the database
- Access to all data table objects
- Table navigation methods
- Table properties and field attributes
- Field data retrieval and assignment
- Lookups
- Sorting
- Support for multiple ACT! databases opened at the same time by a single OLE application

The OLE Database object does not provide functions for the creation of new databases or for scheduling an activity with more than one contact.

Note All objects should be declared and created before use and closed and set to nothing afterwards, implicitly (as in a macro) or explicitly. Using the Close Method before setting objects to null ensures all memory is released and significantly improves performance. Examples in this document illustrate concepts only, and do not always contain all of these steps.

System requirements

Before using the ACT! OLE Database object, the following applications must be already installed on the SDK computer:

- ACT! for Windows, version 3.0.3 or later. We recommend using the latest update. To receive an update, within ACT!, click the **Help** menu, then click **ACT! Update**.
- Microsoft Windows 95/98/2000/Me/XP, or Windows NT 4.0.

Using with Visual C++

Many programmers who work with the Database object work with Microsoft Visual Basic. Hence, most of the examples in the documentation of the database object are designed for the Visual Basic programmer. However, a type library is supplied and consequently, the integrated development environment (IDE) of Microsoft Visual C++ can be used to automatically generate prototypes for the object methods.

The following basic steps can help start use of ACT! automation libraries using Microsoft Visual C++ 4.0 or later and Microsoft Foundation Class (MFC) library. To use the Database object in an application written using Visual C++, first create a wrapper class around the OLE objects supported by the OLE Database object.

To create a wrapper class

1. Create a new MFC project and enable OLE automation.
2. Open the class wizard and click **Add Class**.
3. Select the **From Type Library** option.

4 When prompted for the name of the type-library, choose ACTOLE.TLB from the ACT! program files folder.

5 In the **Confirm Classes** dialog box, click **OK** or select only needed classes, then click **OK**.

Visual C++ automatically generates MFC wrapper classes for all methods and properties supported by the Database object. For more information on how to use OLE automation in Visual C++, see Visual C++ online Help.

For each property, the ACT! Database object creates corresponding sets of Get and Set methods in an MFC wrapper class. Read-only properties have only a Get method. If using Visual C++, check the C++ header file generated by Visual C++ (ACTOLE.H) for the methods that correspond to properties documented in this manual. The following table lists examples of properties and corresponding methods to use in Visual C++.

Property Name	Type	VISUAL C++ Method
Database.ActiveUserCount	Read Only	Database.GetActiveUserCount()
Activity.NextScheduledWith	Read/Write	Activity.GetNextScheduledWith() (to get the value of the property)
		Activity.SetNextScheduledWith() (to get the value of the property)

Sample VISUAL C++ code

```
IDatabase Database;
Database.CreateDispatch("ACTOLE.DATABASE", pException);
Database.Open("c:\\act\\database\\act5demo.dbf");
m_ListBox.ResetContent();
if( Database.GetIsOpen() )
{
     // Attach to the GROUP object.
     IGroup Group;
     LPDISPATCH groupDispatch = Database.GetGroup();
     Group.AttachDispatch(groupDispatch, TRUE);
     // Enumerate all of the GROUPS.
     Group.MoveFirst();
     while( !Group.GetIsEOF() )
     {
        CString szOutput;
        ConvertVariantToString( Group.GetData(GF_Name), szOutput);
          m_ListBox.AddString( szOutput );
        // If there are contacts for this group enumerate them.
        if( Group.GetContactCount() > 0 )
        {
        // Attach to the MEMBERS object.
        IMembers Members;
        LPDISPATCH membersDispatch = Group.GetMembers();
        Members.AttachDispatch(membersDispatch, TRUE);
          Members.MoveFirst();
        while( !Members.GetIsEOF() )
        {
          CString memberString;
          memberString.Format("--> %s", Members.GetName());
          m_ListBox.AddString(memberString);
          Members.MoveNext();
```

```
            }
        }
        Group.MoveNext();
    }
    Group.ReleaseDispatch();
    Database.Close();
    Database.ReleaseDispatch();
}
else
    {
    AfxMessageBox("database did not open");
    }
```

Note This example assumes that the user is logging on a single-user database.

Understanding key files and concepts

The following table lists key files used by the ACT! OLE Database object. These files are stored in the ACT! Program Files folder.

File	Description
ACTOLE.DLL	OLE library containing the ACT! OLE Database object.
ACTOLE.TLB	Type library that contains functions within the ACT! OLE Database object. Methods contained in the type library need to be used directly by Visual C++ developers. In Visual Basic type library functions are handled automatically at run time.
ACTEVENT.OCX	OLE event notification control module, used by third-party applications for receiving event notification. An event is generated, for example, when the contact is changed in the Contact View.
ACTREG.EXE	The ACT! Windows registry update utility. This utility can be run manually if necessary to register ACT! OLE controls as a troubleshooting procedure.

Using the type library

Support is provided for ACT! 3.0, ACT! 4.0, and ACT! 5.0 data access within a 32-bit development environment for ACT! 3.0 or later. The OLE Database object (ACTOLE.DLL) is an OLE automation server with objects implemented as a Windows dynamic link library (DLL) of methods and properties used to simplify communications between OLE programmable languages and an ACT! database. The object is a Visual C++ programmed module using internal buffers to move information to and from the database. To add or change information, fill a buffer with the appropriate data, and then call a method to move the data into the ACT! database. When information from the OLE Database object is requested, the client application returns the buffered data. The ACTOLE.DLL makes calls into the ACT! database layer ADAL.DLL. The ACTOLE.DLL and type library ACTOLE.TLB are installed with the ACT! application and placed in the ACT folder.

The database object provided with ACT! 5.0 is incompatible with the databases from previous ACT! releases. Before using the ACT! 5.0 database object, convert ACT! 3.0 and ACT! 4.0 databases into ACT! 5.0 format.

Unique ID field considerations

ACT! database Unique ID field values are created by calculating a unique value, then modifying it to contain printable characters. The resulting value is a left-justified string, which is padded with enough trailing spaces to complete the 12 character fixed-length Unique ID field. See ACT! Databases for lists of field IDs and names (Field constants).

Caution Do not use the Visual Basic Trim function or any other method to trim the length of the Unique ID string, or the resulting Unique ID will be invalid.

Special data types

This section describes the format for date and time values and phone numbers used by the OLE Database object, which is different from the native format used to store this information in a database.

Date and time formats

The OLE Database object gets and returns date and time values formatted according to Windows Regional Settings Short Date style and Time style. Using the regional settings for the United States, for example, the date and time is formatted as follows:

```
M/d/yy h:mm:ss tt
example: 8/19/97 6:28:29 PM
```

Using the regional settings for Australia, for example, the date and time are formatted as follows:

```
d/MM/yy H:mm:ss
example: 19/08/97 18:28:29
```

Date and time field values must be in the following ranges:

Portion	Description	Range
yyyy	Year	1970 - 2038
MM	Month (January = 01)	01 - 12
dd	Day	01 - 31
hh	Hours after midnight	00 - 23
mm	Minutes after hour	00 - 59
ss	Seconds after minute	00 - 59

Phone formats

The Database object returns phone numbers in the default of the canonical address format, or formatted as displayed in the ACT! application, depending on the value set by the PhoneFormatting property of the Database object.

When setting data in a phone field, the Database object accepts only a TAPI canonical-formatted phone number.

A canonical address is an ASCII string with the following structure:

```
+CountryCode space [(AreaCode) space] SubscriberNumber
```

The phone number string is continuous with the exception of exactly one space after CountryCode and exactly one space after the optional AreaCode.

Element	Definition
+	ASCII Hex (2B). Indicates that the number that follows uses the canonical format.
CountryCode	A variable length string containing only the digits 0-9. It identifies the country in which the address is located.
space	Exactly one ASCII space character (0x20). The space is used to delimit the end of the CountryCode part of an address.
[(*AreaCode*) space]	A variable length string containing only the digits 0-9. The optional AreaCode is the area code part of the address. If present, it must enclosed in parentheses and followed by a space as a delimiter.
SubscriberNumber	A variable length string containing the digits 0-9 and formatting characters, including any of the following dialing control characters: AaBbCcDdPpTtWw*#!,@$? The subscriber number cannot contain the left parenthesis or right parenthesis character (which is used only to delimit the area code), and cannot contain the "|", "^", or CRLF characters, which are used to begin the following fields. Most non-digit characters in the subscriber number are spaces, periods ("."), and dashes ("-").

The following is an example of an Australian phone number in TAPI canonical format, with a breakdown of its different parts:

```
+61 (3) 8236887
```

Country Code 61 (Australia)

Area Code 3

Subscriber Number 8236887

For a complete list of country codes and default formats look at the PHONE.FMT file. This file is installed in the ACT folder.

Caution Do not modify the PHONE.FMT file.

Error Codes

The following error codes apply only to the ACT! application. If an error occurs that is not caused by the OLE automation client's code, please record the return value and the situation that caused it for use by technical support.

Value	Error code	Description
-2000	Status_Obsolete	for obsolete calls
-1999	Status_InvalidParam	invalid parameter
-1500	Status_SecurityLocked	the security is currently locked
-1499	Status_SecurityEditFailed	general error trying to edit security
-1000	Status_CantDeleteUser	cannot delete a user record
-750	Status_CantEdit	cannot edit
-500	Status_DataLocked	the record is locked
-499	Status_DataNotLocked	the record is not locked

Value	Error code	Description
-498	Status_DataEditNow	the record is being edited
-497	Status_DataUninitialized	the record has not been initialized
-496	Status_ListUninitialized	the table has not been initialized
-495	Status_DataDeleted	the record has been deleted
-494	Status_DataChanged	the record has changed
-493	Status_CantLockData	the record cannot be locked
-109	Status_ExceedLimit	limit has been exceeded
-108	Status_DBAccessDenied	database access is denied
-107	Status_CantEditData	the record cannot be edited
-106	Status_InvalidFieldType	invalid field type
-105	Status_InvalidSortField	invalid sort field
-104	Status_OutOfRange	the record/field is out of range
-103	Status_InvalidBufIndex	invalid buffer index
-102	Status_DataBeingEdited	the record is currently being edited
-101	Status_CantDeleteField	the field cannot be deleted
-100	Status_InvalidSchemaInfo	invalid schema information
-99	Status_InvalidFieldName	invalid field name
-98	Status_InvalidField	invalid field
-97	Status_CantEditAttr	the attribute cannot be edited
-96	Status_ExceedsMaxLength	maximum length exceeded
-95	Status_CantEditSchema	the schema information cannot be edited
-94	Status_DupeFieldName	duplicate field name
-93	Status_CantEditSecurity	the security information cannot be edited
-92	Status_InvalidId	invalid ID specified
-91	Status_DBSchemaCorrupt	the schema information is corrupted
-90	Status_DBDataCorrupt	the data file is corrupted
-89	Status_DBIndexCorrupt	the index file is corrupted
-88	Status_DBCreateFailed	create failed
-87	Status_DBOpenFailed	open failed
-86	Status_DBLockFailed	lock failed
-82	Status_IOFailed	read/write failed
-80	Status_CreateFailed	create failed
-79	Status_OpenFailed	open failed
-78	Status_DBInvalidVersion	invalid version to open database
-77	Status_Duplicate	duplication occurred
-76	Status_InvalidPrivilege	invalid permission
-75	Status_SharingViolation	sharing violation
-50	Status_InvalidFilename	filename invalid or not found

Value	Error code	Description
-10	Status_DBLocked	the database is locked
-9	Status_DBOpen	the database is already open
-8	Status_DBAlreadyOpen	the database is already open
-7	Status_DBNotOpen	the database is not open
-6	Status_FatalError	a fatal error occurred
-5	Status_OutofDiskSpace	insufficient disk space
-4	Status_Unimplemented	not implemented
-3	Status_OutofMemory	insufficient memory
-2	Status_Unsupported	feature not supported
-1	Status_GeneralError	a general error occurred
0	Status_Success	success
1	Status_Supported	supported
2	Status_NoMoreData	no more data available
3	Status_NoData	no data available
4	Status_NoCount	no count available
5	Status_PartialCount	partial count only available
6	Status_NoPosition	no position (BOF/EOF)
7	Status_PartialPosition	partial position
8	Status_Cancel	canceled

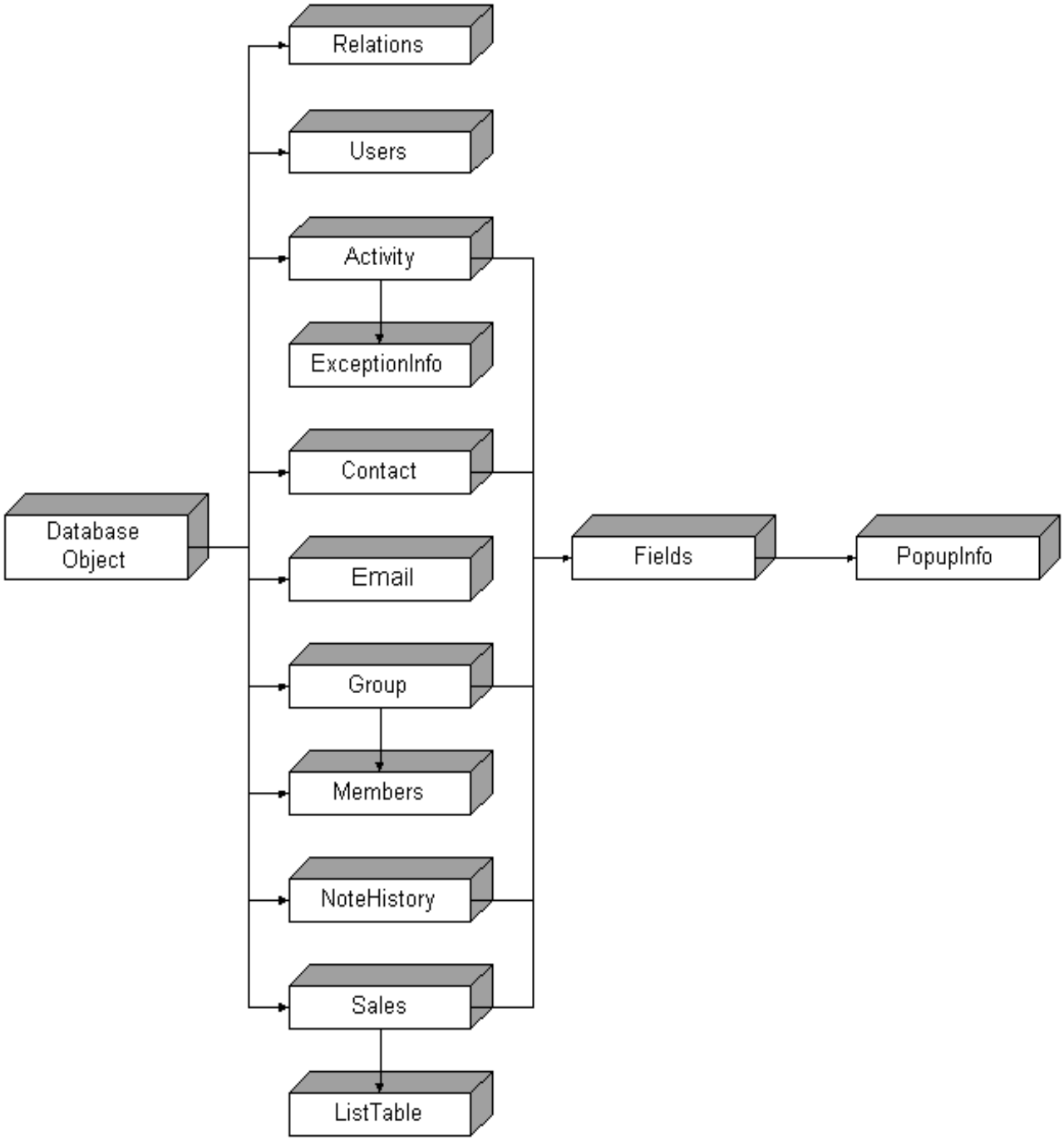
Object definitions

The Database class has the following derived objects:

Name	Definition
Activity object	Contains scheduled activity information for the active Database object.
Contact object	Contains information about the ACT! contacts.
Database object	Is the OLE creatable object that contains the ACT! database information and objects. This is the programmable interface for Activity, Contact, Group, and NoteHistory objects.
Email object	Contains e-mail information for the active database object.
ExceptionInfo object	Contains the recurring exception information for a particular activity.
Fields object	Contains the data properties for a specified ACT! field.
Group object	Contains group and membership information for the active Database object.
ListTable object	Contains product, product type, and main competitor data for use by the Sales object in ACT! 5.0 or later.
Members object	Contains membership information for the active Group object.
NoteHistory object	Contains notes and historical information for the active Database object.
PopupInfo object	Contains the popup information for a valid field.
Relations object	Contains relational information between table objects of the active Database object.
Sales object	Contains data for sales process management for the active Database object in ACT! 5.0 or later.
Users object	Contains the user information for the active Database object.

Database object model

The following shows the relationships of objects in the Database object model:



Common properties and methods

The following common properties and methods apply to more than one ACT! database object. The database objects to which the property or method applies are listed in the following tables.

Properties

Name	Objects	Parameter(s)	Parameter type(s)	Value type	Access
Data Property	Activity, Contact, E-mail, Group, ListTable, Members, NoteHistory, Sales	*iFieldID[, szValue]*	Short Integer, String	Date/Variant	Read/Write
Error Property	Activity, Contact, Database, E-mail, ExceptionInfo, Fields, Group, ListTable, Members, NoteHistory, PopupInfo, Relations, Sales, Users	*None*		Boolean	Read Only
FieldCount Property	Activity, Contact, E-mail, Group, ListTable, Members, NoteHistory, Sales	*None*		Short Integer	Read Only
Fields Property	Activity, Contact, E-mail, Group, ListTable, Members, NoteHistory, Sales	*None*		Object/LPDISPATCH	Read Only
IsBOF Property	Activity, Contact, E-mail, Group, ListTable, Members, NoteHistory, Sales	*None*		Boolean	Read Only
IsEOF Property	Activity, Contact, E-mail, Group, ListTable, Members, NoteHistory, Sales	*None*		Boolean	Read Only
IsLocked Property	Activity, Contact, E-mail, Group, ListTable, NoteHistory, Sales	*None*		Boolean	Read Only
IsOpen Property	Activity, Contact, E-mail, Group, ListTable, NoteHistory, Sales	*None*		Boolean	Read Only
LastError Property	Activity, Contact, Database, E-mail, ExceptionInfo, Fields, Group, ListTable, Members, NoteHistory, PopupInfo, Relations, Sales, Users	*None*		Short Integer	Read Only
LockLevel Property	Activity, Contact, E-mail, Group, ListTable, NoteHistory, Sales	*None*		Short Integer	Read Only
Name Property	Activity, Contact, E-mail, Group, ListTable, Members, NoteHistory, Sales	*None*		String	Read Only
Position Property	Activity, Contact, E-mail, Group, ListTable, Members, NoteHistory, Sales	*None*		Long Integer	Read Only
Query Property	Activity, Contact, E-mail, Group, ListTable, Members, NoteHistory, Sales	*szString*	String	String	Read/Write
RecordCount Property	Activity, Contact, E-mail, Group, ListTable, Members, NoteHistory, Sales	*None*		Long Integer	Read Only

Data Property

Description Retrieves or sets data for a specified field in the specified object.

Objects Activity object, Contact object, Email object, Group object, Members object, NoteHistory object

Also applies to the ListTable object and Sales object in ACT! 5.0 or later.

Syntax *object*.**Data** *iFieldID [, szValue]*

Parameters *iFieldID* A short integer that specifies the unique field ID for the specified table. See ACT! Databases for lists of field IDs and names (Field constants).

[*szValue*] A string that specifies the data to set for the specified field. Omit this optional parameter to get the data in the field.

Value type Date/Variant, Read/Write

Comments To set data in the database, call Add or Edit before using the Data property. Follow the Data assignment with a call to Update to commit the changes to the database. If you call a navigation commend, such as MoveFirst, before Update, any changes are lost.

All field data is passed and retrieved in string format only. Use the Type property in the Fields object to get the required data type for the data.

See also Add Method, Edit Method, FieldCount Property, IsBOF Property, IsEOF Property, RecordCount Property, Update Method

Type Property in Fields object

Example

```
'This example demonstrates how to retrieve data from a record.

dim objDatabase as object
Set objDatabase = CreateObject("ACTOLE.DATABASE")
objDatabase.Open dbName

Dim tableObject As Object
Set tableObject = objDatabase.CONTACT

tableObject.MoveFirst

'Get the name of the contact
dim strOutput as String
strOutput = tableObject.Data(CF_Name)

set tableObject = Nothing

objDatabase.Close
set objDatabase = Nothing
```

Error Property

Description Returns the error status of True for the object after a call that generated an error. Returns False if no error was generated on the last call. The error status is set to False at the beginning of each operation. If an error status of True is returned, call LastError to get the error code. See Error Codes for error code descriptions.

Objects Activity object, Contact object, Database object, Email object, ExceptionInfo object , Fields object, Group object, Members object, NoteHistory object , PopupInfo object, Relations object, Users object

Also applies to the ListTable object and Sales object in ACT! 5.0 or later.

Syntax *object.***Error**

Value type Boolean, Read Only

Comments The error property resets to False at the beginning of every operation, so check for errors after each function that could generate an error for accurate error checking.

See also LastError Property

FieldCount Property

Description Returns the number of fields in the table object. Use this value to iterate through the Data property and Fields property.

Objects Activity object, Contact object, Email object, Group object, Members object, NoteHistory object

Also applies to the ListTable object and Sales object in ACT! 5.0 or later.

Syntax *object.***FieldCount**

Value type Short Integer, Read Only

See also Data Property, IsBOF Property, IsEOF Property, RecordCount Property

Fields Property

Description Returns a Fields object. The Fields object returned by this property is directly associated with its parent object. For example, if the parent object is Contact, then the Fields object is initialized with the fields for the Contact table.

Objects Activity object, Contact object, Email object, Group object, Members object, NoteHistory object

Also applies to the ListTable object and Sales object in ACT! 5.0 or later.

Syntax *object.***Fields**

Value type Object/LPDISPATCH, Read Only

See also Fields object

Example

```
'This example demonstrates how to retrieve fields object from a list object.

dim objDatabase as object
Set objDatabase = CreateObject("ACTOLE.DATABASE")
objDatabase.Open dbName

'Obtain the Fields object for the Contact object
dim fields as object
set fields = objDatabase.CONTACT.FIELDS

objDatabase.Close
set objDatabase = Nothing
```

IsBOF Property

Description Returns True if the current record is the first record in the table or False if the current record is not the first record in the table.

Objects Activity object, Contact object, Email object, Group object, Members object, NoteHistory object

Also applies to the ListTable object and Sales object in ACT! 5.0 or later.

Syntax *object*.**IsBOF**

Value type Boolean, Read Only

See also RecordCount Property

IsEOF Property

Description Returns True if an attempt has been made to move past the last record in the table or False if the current record is not the last record in the table.

Objects Activity object, Contact object, Email object, Group object, Members object, NoteHistory object

Also applies to the ListTable object and Sales object in ACT! 5.0 or later.

Syntax *object*.**IsEOF**

Value type Boolean, Read Only

See also RecordCount Property

IsLocked Property

Description Returns True if the current database is locked or False if it is not locked. Call this property immediately before performing database functions, such as Reindex, or before using batch operations..

Objects Activity object, Contact object, Email object, Group object, NoteHistory object

Also applies to the ListTable object and Sales object in ACT! 5.0 or later.

Syntax *object*.**IsLocked**

Value type Boolean, Read Only

Comments Use LockLevel to determine if the current record is unlocked or has been locked by the current user or by a network user.

See also Add Method, Edit Method, GoTo Method, LockLevel Property, MoveFirst Method, MoveLast Method, MoveNext Method, MovePrevious Method

Example See BeginBatchInsert Method in the Database object.

IsOpen Property

Description Returns True if the current table object is open or False if it is not open. Call this property to determine if a database has been successfully opened.

Objects Activity object, Contact object, Email object, Group object, NoteHistory object

Also applies to the ListTable object and Sales object in ACT! 5.0 or later.

Syntax *object*.**IsOpen**

Value type Boolean, Read Only

See also Close Method,

Open Method in Database object

LastError Property

Description Returns the last error code for the object. This status should be checked if Error is True to determine the cause of the last error. After checking this property, the last error status is reset to Status_Success. See Error Codes.

Objects Activity object, Contact object, Database object, Email object, ExceptionInfo object , Fields object, Group object, Members object, NoteHistory object , PopupInfo object, Relations object, Users object

Also applies to the ListTable object and Sales object in ACT! 5.0 or later.

Syntax *object.***LastError**

Value type Short Integer, Read Only

See also Error Property

LockLevel Property

Description Returns record lock status 0 if the current record is not locked, 1 if locked by the current user, or 2 if locked by a network user. Call this property prior to calling Edit to ensure an update to the record is possible.

Objects Activity object, Contact object, Email object, Group object, NoteHistory object

Also applies to the ListTable object and Sales object in ACT! 5.0 or later.

Syntax *object.***LockLevel**

Value type Short Integer, Read Only

Comments Use the IsLocked Property to determine if the entire database is locked.

See also Add Method, Edit Method, GoTo Method, IsLocked Property, MoveFirst Method, MoveLast Method, MoveNext Method, MovePrevious Method

Name Property

Description Returns the name of the current table object. Use this property to determine which table is the current object.

Objects Activity object, Contact object, Email object, Group object, Members object, NoteHistory object

Also applies to the ListTable object and Sales object in ACT! 5.0 or later.

Syntax *object.***Name**

Value type String, Read Only

Position Property

Description Returns the current record position in a table.

Objects Activity object, Contact object, Email object, Group object, Members object, NoteHistory object

Also applies to the ListTable object and Sales object in ACT! 5.0 or later.

Syntax *object.***Position**

Value type Long Integer, Read Only

Query Property

Description Gets and sets the string for an ACT!-formatted query.

Objects Activity object, Contact object, Group object, Members object, NoteHistory object

Also applies to the ListTable object and Sales object in ACT! 5.0 or later.

Syntax *object*.**Query** = "*Contact Contains ""szString"""*

or

object.**Query** = "*Group Contains ""szString"""*

Parameters *szString* A string that specifies the query criteria.

Value type String, Read/Write

Comments Queries start by first resetting the scope to all records, then executing the query. After executing, a query returns a subset of records. To clear an existing query and reset the scope to all records, pass in an empty string, then call the Execute method. See the ACT! online Help for information on advanced queries. The advanced Query Helper can also create and test query strings. Unique ID type fields cannot be used in queries.

See also Execute Method

Example See the Execute method.

RecordCount Property

Description Returns the number of records in a table.

Objects Activity object, Contact object, Email object, Group object, Members object, NoteHistory object

Also applies to the ListTable object and Sales object in ACT! 5.0 or later.

Syntax *object*.**RecordCount**

Value type Long Integer, Read Only

Example See Jump Method.

Methods

Name	Objects	Parameter(s)	Parameter type(s)	Return type
Add Method	Activity, Contact, Group, ListTable, NoteHistory, Sales	None		Void
Close Method	Activity, Contact, Group, ListTable, NoteHistory, Sales	None		Void
Delete Method	Activity, Contact, Group, ListTable, NoteHistory, Sales	None		Void
Edit Method	Activity, Contact, Group, ListTable, NoteHistory, Sales	None		Void
Execute Method	Activity, Contact, Group, ListTable, Members, NoteHistory, Sales	None		Void
FindDuplicates Method	Contact, Group	None		Void
GetDataEx Method	Activity, Contact, Group, ListTable, NoteHistory, Sales	iFieldArray, szValueArray	Short Integer, String	Void
GetDuplicateCriteria Method	Contact, Group	iField1ID, iField2ID, iField3ID	Short Integer, Short Integer, Short Integer,	Short Integer

Name	Objects	Parameter(s)	Parameter type(s)	Return type
GetSort Method	Activity, Contact, Group, ListTable, Members, NoteHistory, Sales	iField1ID, iDirection1, iField2ID, iDirection2, iField3ID, iDirection3	Short Integer, Short Integer, Short Integer, Short Integer, Short Integer, Short Integer	Short Integer
GoTo Method	Activity, Contact, Group, ListTable, Members, NoteHistory, Sales	szUniqueID	String	Void
Jump Method	Activity, Contact, Group, ListTable, Members, NoteHistory, Sales	lValue	Long Integer	Void
Lookup Method	Activity, Contact, Group, ListTable, Members, NoteHistory, Sales	iFieldID, szKeyword, iLookupType	Short Integer, String, Short Integer	Void
LookupKeyword Method	Contact, Group, NoteHistory	szKeyword	String	Void
MoveFirst Method	Activity, Contact, Group, ListTable, Members, NoteHistory, Sales	None		Void
MoveLast Method	Activity, Contact, Group, ListTable, Members, NoteHistory, Sales	None		Void
MoveNext Method	Activity, Contact, Group, ListTable, Members, NoteHistory, Sales	None		Void
MovePrevious Method	Activity, Contact, Group, ListTable, Members, NoteHistory, Sales	None		Void
Rebuild Method	Activity, Group, ListTable, Members, NoteHistory, Sales	None		Void
SetDataEx Method	Activity, Contact, Group, ListTable, NoteHistory, Sales	iFieldArray szValueArray	Short Integer String	Void
SetDuplicateCriteria Method	Contact, Group	iFieldID1 iFieldID2 iFieldID3	Short Integer, Short Integer, Short Integer	Void
Sort Method	Activity, Contact, Group, ListTable, Members, NoteHistory, Sales	iField1ID iDirection1 iField2ID iDirection2 iField3ID iDirection3	Short Integer, Short Integer, Short Integer, Short Integer, Short Integer, Short Integer	Void
Update Method	Activity, Contact, Group, ListTable, Members, NoteHistory, Sales	None		String

Add Method

Description Adds a new record with default field values in the current object.

Note: Use the AddSubGroup Method in the Group object to add a subgroup record.

Objects Activity object, Contact object, Group object, NoteHistory object

Also applies to the ListTable object and Sales object in ACT! 5.0 or later.

Syntax *object.***Add**

Comments Creates a new record buffer with default field values so that field data can be assigned using the Data property. Added records are not committed until the Update method is called. If any navigational commands are used prior to update, the new record is lost.

See also Data Property, Delete Method, GoTo Method, MoveFirst Method, MoveLast Method, MoveNext Method, MovePrevious Method, Update Method

FirstScheduledWith Property, NextScheduledWith Property in Activity object

Example

```
'This example demonstrates how to add a new contact record.

dim objDatabase as object
dim szUniqueId as string
Set objDatabase = CreateObject("ACTOLE.DATABASE")
objDatabase.Open dbName

objDatabase.CONTACT.Add
'Set field data
objDatabase.CONTACT.Data CF_Company, "WhatCo"
objDatabase.CONTACT.Data CF_Name, "Chris Huffman"
objDatabase.CONTACT.Data CF_Address1, "20323 SomeWhere Avenue"
objDatabase.CONTACT.Data CF_Title, "President"

'Update the records contents
szUniqueId = objDatabase.CONTACT.Update
objDatabase.Contact.GoTo (szUniqueId)

objDatabase.Close
set objDatabase = Nothing
```

Close Method

Description Closes the current table. The table schema and record information is discarded. To improve performance by ensuring that all memory is released, always use this method before setting the object to null.

Objects Activity object, Contact object, Group object, NoteHistory object

Also applies to the ListTable object and Sales object in ACT! 5.0 or later.

Syntax *object.***Close**

See also IsOpen Property

Open Method in Database object

Delete Method

Description Deletes the currently selected record.

Caution: This is a single step process that cannot be reversed. Before deleting a parent group record, delete its subgroup records.

Objects Activity object, Contact object, Group object, NoteHistory object

Also applies to the ListTable object and Sales object in ACT! 5.0 or later.

Syntax *object.***Delete**

Return type Void

See also Add Method, GoTo Method, IsLocked Property, LockLevel Property, MoveFirst Method, MoveLast Method, MoveNext Method, MovePrevious Method, Update Method

Edit Method

Description Prepares the current record for editing and applies a record lock.

Objects Activity object, Contact object, Group object, NoteHistory object

Also applies to the ListTable object and Sales object in ACT! 5.0 or later.

Syntax *object.***Edit**

Return type Void

Comments This method enables the current record buffer to be edited and sets default field values so that field data can then be assigned using the Data property. Edited records are not committed until the Update method is called. If any navigational commands are used prior to update, the modifications are lost. To ensure the edit will succeed, call IsLocked and LockLevel immediately before calling this method.

See also Data Property, GoTo Method, IsLocked Property, LockLevel Property, MoveFirst Method, MoveLast Method, MoveNext Method, MovePrevious Method, Update Method

Example

```
'This example demonstrates how to edit an existing contact record.
dim objDatabase as object
Set objDatabase = CreateObject("ACTOLE.DATABASE")
objDatabase.Open dbName

objDatabase.CONTACT.MoveFirst
objDatabase.CONTACT.Edit

'Change the address
objDatabase.CONTACT.Data CF_Address1, "20323 Somewhere Avenue"

'Update the records contents
objDatabase.CONTACT.Update

objDatabase.Close
set objDatabase = Nothing
```

Execute Method

Description Executes a query that has been set by the Query property and clears any previous query.

Objects Contact object, Group object,

Also applies to Activity object, Members object, NoteHistory object in ACT! 3.0.7 or later.

Also applies to the ListTable object and Sales object in ACT! 5.0 or later.

Syntax *object.***Execute**

Return type Void

See also Query Property

Example

```
'This example demonstrates how to use queries.

dim objDatabase as object
Set objDatabase = CreateObject("ACTOLE.DATABASE")
objDatabase.Open dbName

objDatabase.CONTACT.Query = "CONTACT contains ""Smith"""
objDatabase.CONTACT.Execute

'*** Can now retrieve data for record(s)
objDatabase.Close
set objDatabase = Nothing
```

FindDuplicates Method

Requires ACT! 5.0 or later

Description Uses the duplicate checking criteria to search for duplicate contacts and groups in the Contact table or Group table. After executing this command, the Current lookup contains the duplicates.

Objects Contact object, Group object

Syntax *object*.**FindDuplicates**

Return type Void

See also GetDuplicateCriteria Method, SetDuplicateCriteria Method

Example

```
'The following code finds duplicates and then enumerates through the
'duplicate records.

Dim objDatabase as object
Set objDatabase= CreateObject("ACTOLE.DATABASE")
objDatabase.Open dbName

'Assign the correct table object
Set objContact = objDatabase.Contact
If objDatabase.IsOpen Then
  'Find the duplicate contacts in this database
  objContact.FindDuplicates
  List1.AddItem objContact.recordCount & " duplicate contacts Found!"

  'Enumerate all the duplicates in the contact table
  objContact.MoveFirst
  While Not objContact.IsEOF
     List1.AddItem "CONTACT: " & objContact.Data(CF_Name)
     List1.AddItem "TITLE: " & objContact.Data(CF_Title)
     List1.AddItem "COMPANY: " & objContact.Data(CF_Company)
     List1.AddItem objContact.Data(CF_City) & ", " &
     objContact.Data(CF_State)
     objContact.MoveNext
  Wend
  Set objContact = Nothing
  objDatabase.close
End If
Set objDatabase = Nothing
```

GetDataEx Method

Requires ACT! 4.0.2 or later

Description Returns an array of strings containing the data for each field specified in an array of fields.

Objects Activity object, Contact object, Group object, NoteHistory object

Also applies to the ListTable object and Sales object in ACT! 5.0 or later.

Syntax *object*.**GetDataEx** *iFieldArray, szValueArray*

Parameters *iFieldArray* Returns an array of short integers (or pointers to short integers in VISUAL C++) that represent the field IDs of fields for which to get the data contained in the fields. The Field ID array must be an array of integers - variant arrays do not work.

szValueArray Returns an array of strings (or BSTRs in VISUAL C++) for the data that is returned for the specified fields. The same number of elements must be specified for the iFieldArray and the szValueArray parameters.

Note: In Visual C++, use VARIANT.pparray instead of VARIANT.parray for pointers to elements in the array.

Return type Void

See also SetDataEx Method

Example

```
'This example demonstrates getting data from multiple fields and sending
'it to a file.

Dim NoteIdArray(10) As Integer
Dim OutputArray(10) As String
Dim objDatabase As Object
Dim objNoteHistory As Object
Dim count As Double
Dim LastRecord As Long
Dim FileNum

NoteIdArray(0) = NHF_ContactId'Field ID constant
NoteIdArray(1) = NHF_UserTime'Include Actfield.bas
NoteIdArray(2) = NHF_Type
NoteIdArray(3) = NHF_Text

Set objDatabase = CreateObject("ACTOLE.DATABASE")
objDatabase.Open dbName
If objDatabase.IsMultiUser Then
  objDatabase.ValidateUser "Chris Huffman", ""
End If
If objDatabase.IsOpen = False Then
  MsgBox "Failed to open database"
End If

Set objNoteHistory = objDatabase.NoteHistory
LastRecord = objNoteHistory.recordcount
count = 1
'Get a free file number
FileNum = FreeFile
'Open GetNoteHistory.TXT for append
Open "GetNoteHistory.txt" For Append As FileNum
objNoteHistory.MoveFirst

Do While (count <= LastRecord)

  objNoteHistory.GetDataEx NoteIdArray, OutputArray

  Write #FileNum, OutputArray(0), OutputArray(1), OutputArray(2),
    OutputArray(3)
  count = count + 1
  objNoteHistory.MoveNext
  If objNoteHistory.IsEOF Then
     Exit Do
  End If
Loop
```

```
Close FileNum
'Close the file
Set objNoteHistory = Nothing

'Clean up
objDatabase.Close
Set objDatabase = Nothing
```

GetDuplicateCriteria Method

Requires ACT! 5.0 or later

Description Returns short integer values for the field IDs for the duplicate checking criteria for the Contact table or Group table.

Object Contact object, Group object

Syntax *object*.**GetDuplicateCriteria** (*iFieldID1, iFieldID2, iFieldID3*)

Parameters *iFieldID1* A short integer (or pointer to a short integer in VISUAL C++) that represents the field ID type for the first field for duplicate checking (find a match for data in this field).

iFieldID2 A short integer (or pointer to a short integer in VISUAL C++) that represents the field ID type for the second field for duplicate checking (then find a match for data in this field).

iFieldID3 A short integer (or pointer to a short integer in VISUAL C++) that represents the field ID type for the third field for duplicate checking (then find a match for data in this field).

See ACT! Databases for lists of field IDs and names (Field constants).

See also FindDuplicates Method , SetDuplicateCriteria Method

Example

```
Dim objDatabase as object
Set objDatabase = CreateObject("ACTOLE.DATABASE")'Open the database
objDatabase.Open dbName

'Assign the correct table object
Set objContact = objDatabase.Contact

If objDatabase.IsOpen Then
  'Get the current duplicate criteria
  objContact.GetDuplicateCriteria DupCriteria1, DupCriteria2, DupCriteria3
  List1.AddItem "Duplicate criteria is :  Match on   " & DupCriteria1 & "
    and then on  " & DupCriteria2 & "  and then on " & DupCriteria3
'Set the duplicate criteria as match on User1 then Department and then
'on Title
  objContact.SetDuplicateCriteria CF_User1, CF_Department, CF_Title
  Set objContact = Nothing
  objDatabase.close
End If

Set objDatabase = Nothing
```

GetSort Method

Requires ACT! 4.0 or later

Description Returns short integer values for the field IDs and sort directions of the sorted fields for the active sort of an ACT! database table. A value of 0, 1, 2, or 3 is returned specifying the number of fields sorted for the table.

Objects Activity object, Contact object, Group object, NoteHistory object

Also applies to the ListTable object and Sales object in ACT! 5.0 or later.

Syntax *object*.**GetSort** *iField1ID, iDirection1, iField2ID, iDirection2, iField3ID, iDirection3*

Parameters *iField1ID* A short integer (or pointer to a short integer in VISUAL C++) that represents the field ID for the first field in the sort order.

iDirection1 A short integer (or pointer to a short integer in VISUAL C++) that represents the sort direction for the first field. Values returned are 0 for ascending order or 1 for descending order.

iField2ID A short integer (or pointer to a short integer in VISUAL C++) that represents the field ID for the second field in the sort order.

iDirection2 A short integer (or pointer to a short integer in VISUAL C++) that represents the sort direction for the second field. Values returned are 0 for ascending order or 1 for descending order.

iField3ID A short integer (or pointer to a short integer in VISUAL C++) that represents the field ID for the third field in the sort order.

iDirection3 A short integer (or pointer to a short integer in VISUAL C++) that represents the sort direction for the third field. Values returned are 0 for ascending order or 1 for descending order.

See ACT! Databases for lists of field IDs and names (Field constants).

Return type Short Integer

See also Sort Method

IsSortable Property in Fields object

Example '

```
This example gets the sorted fields in the contact table.

Dim i As Integer
Dim objDatabase as object
Dim iSortFld1, iSortFld2, iSortFld3 as integer
Dim iSortOrder1, iSortOrder2, iSortOrder3 as integer
Set objDatabase = CreateObject("ACTOLE.DATABASE")
objDatabase.Open dbName

'Assign the correct table object
Set objContact = objDatabase.CONTACT
  If objDatabase.IsOpen Then
     i = objContact.GetSort(iSortFld1, iSortOrder1, iSortFld2, iSortOrder2,
       iSortFld3, iSortOrder3)
     lstDisplay.AddItem "i = " & i & "  Sort Order = " & iSortFld1 & " ,"
       & iSortOrder1 & " ," &  iSortFld2 & " ," & iSortOrder2 & " ," &
       iSortFld3 & " ," & iSortOrder3
     Set objContact = Nothing
     objDatabase.Close
  End If
Set objDatabase = Nothing
```

GoTo Method

Description Goes to the specified record and makes it the current record, if it exists within the active lookup.

Objects Activity object, Contact object, Group object, Members object, NoteHistory object

Also applies to the ListTable object and Sales object in ACT! 5.0 or later.

Syntax *object*.**GoTo** *szUniqueID*

Parameters *szUniqueID* A string that specifies the Unique ID of the record.

Return type Void

See also Data Property

UniqueId Property in Users object

Jump Method

Description Goes to a specified position within the table.

Objects Activity object, Contact object, Group object, Members object, NoteHistory object

Also applies to the ListTable object and Sales object in ACT! 5.0 or later.

Syntax *object*.**Jump** *lValue*

Parameters *lValue* A long integer that specifies the new position in the list, in the range between 1 and RecordCount.

Return type Void

See also Position Property, RecordCount Property

Example

```
'This example demonstrates how to jump by position within a table.

dim objDatabase as object
Set objDatabase = CreateObject("ACTOLE.DATABASE")
objDatabase.Open dbName

'Set the current position to the fifth record. assume there
'are more than 5 records.
objDatabase.CONTACT.Jump 5

'Can now retrieve record
objDatabase.Close
set objDatabase = Nothing
```

Lookup Method

Description Performs a lookup for the specified keyword. The lookup is restricted to searching on one field in a table at a time. An error is generated if the specified field does not exist in the table.

Note: This method was modified for the NoteHistory object in ACT! 4.0 or later to add the capability of searching for text in the Regarding field of Notes/History table records.

Objects Activity object, Contact object, Group object, Members object, NoteHistory object

Also applies to the ListTable object and Sales object in ACT! 5.0 or later.

Syntax *object*.**Lookup** *iFieldID, szKeyword, iLookupType*

Parameters *iFieldID* A short integer that specifies the field ID of the field on which the lookup will be performed. See ACT! Databases for lists of field IDs and names (Field constants).

szKeyword A string that specifies the search criteria.

iLookupType A short integer that specifies the type of lookup. Specify 1 to replace the lookup, 2 to add to the lookup, or 3 to narrow the lookup.

Return type Void

Comments Lookups are not exact searches. Instead, they treat the keyword like a Begins With search. For example, if the keyword "B" is specified on the Contact field, the lookup will return all records whose Contact field begins with a B.

See also Error Property, IsBOF Property, LastError Property, RecordCount Property

Example

```
'This example demonstrates how to perform a lookup on the contact table.

dim objDatabase as object
Set objDatabase = CreateObject("ACTOLE.DATABASE")
objDatabase.Open dbName

'Perform Lookup - will return all records whose name begins with Chris
objDatabase.CONTACT.Lookup CF_Name, "Chris", 1

objDatabase.Close
set objDatabase = Nothing
```

LookupKeyword Method

Description Looks up the specified keyword and returns the records containing the keyword.

Objects Contact object, Group object, NoteHistory object

Syntax *object*.**LookupKeyword** *szKeyword*

Parameters *szKeyword* A string specifying the keyword.

Return type Void

Example

```
'The following code does a keyword search for "SDK " and then enumerates
'through all the Contact records that meet this criteria.

Dim objDatabase as object
Set objDatabase= CreateObject("ACTOLE.DATABASE")
objDatabase.Open dbName

'Assign the correct table object
Set objContact = objDatabase.Contact
If objDatabase.IsOpen Then
  'Look up all Contact records in which there is "SDK" in the
  'Contact table.
  objContact.lookupKeyword "SDK"
  List1.AddItem objContact.recordCount & "  contacts Found!"

  'Enumerate all the duplicates in the Contact table
  objContact.MoveFirst
```

```
  While Not objContact.IsEOF
     List1.AddItem "CONTACT: " & objContact.Data(CF_Name)
     List1.AddItem "TITLE: " & objContact.Data(CF_Title)
     List1.AddItem "COMPANY: " & objContact.Data(CF_Company)
     List1.AddItem objContact.Data(CF_City) & ", " &
     objContact.Data(CF_State)
     objContact.MoveNext
  Wend
  Set objContact = Nothing
  objDatabase.close
End If

Set objDatabase = Nothing
```

MoveFirst Method

Description Goes to the first record in a table.

Objects Activity object, Contact object, Group object, Members object, NoteHistory object

Also applies to the ListTable object and Sales object in ACT! 5.0 or later.

Syntax *object*.**MoveFirst**

Return type Void

See also MoveLast Method, MoveNext Method, MovePrevious Method

Example See RecordCount Property.

MoveLast Method

Description Goes to the last record in a table.

Objects Activity object, Contact object, Group object, Members object, NoteHistory object

Also applies to the ListTable object and Sales object in ACT! 5.0 or later.

Syntax *object*.**MoveLast**

Return type Void

See also MoveFirst Method, MoveNext Method, MovePrevious Method, RecordCount Property

MoveNext Method

Description Goes to the next record in the table. An error is not generated for an EOF condition. Call IsEOF after calling this method.

Objects Activity object, Contact object, Group object, Members object, NoteHistory object

Also applies to the ListTable object and Sales object in ACT! 5.0 or later.

Syntax *object*.**MoveNext**

Return type Void

See also IsEOF Property, MoveFirst Method, MoveLast Method, MovePrevious Method, RecordCount Property

Example

```
tableObject.MoveNext
If tableObject.IsEOF Then
  MsgBox "End of table"
End If
```

MovePrevious Method

Description Goes to the previous record in the table. An error is not generated for a BOF condition. Call IsBOF after calling this method.

Objects Activity object, Contact object, Group object, Members object, NoteHistory object

Also applies to the ListTable object and Sales object in ACT! 5.0 or later.

Syntax *object*.**MovePrevious**

Return type Void

See also IsBOF Property, MoveFirst Method, MoveLast Method, MoveNext Method, RecordCount Property

Rebuild Method

Description Rebuilds a table, which reloads the table information. Rebuild a table after modifying the table definitions.

Objects Activity object, Group object, Members object, NoteHistory object

Also applies to the ListTable object and Sales object in ACT! 5.0 or later.

Syntax *object*.**Rebuild**

Return type Void

SetDataEx Method

Requires ACT! 4.0.2 or later

Description Sets specified values for each field specified in an array of fields.

Objects Activity object, Contact object, Group object, NoteHistory object

Also applies to the ListTable object and Sales object in ACT! 5.0 or later.

Syntax *object*.**SetDataEx** *iFieldArray, szValueArray*

Parameters *iFieldArray* An array of short integers (or pointers to short integers in VISUAL C++) that represent the field IDs of fields for which to set the specified data. See ACT! Databases for lists of field IDs and names (Field constants).

szValueArray An array of strings (or BSTRs in VISUAL C++) for the data to set to the specified fields. The same number of elements must be specified for the iFieldArray and the szValueArray parameters.

Note: In Visual C++, use VARIANT.pparray instead of VARIANT.parray for pointers to elements in the array.

Return type Void

See also GetDataEx Method

Example

```
'This example demonstrates updating multiple fields in one operation.

Dim database As Object
Dim contact As Object
Dim fieldarray(2) As Integer  'Must be a short integer
Dim dataarray(2) As String    'Must be a string (BSTR)

'Open the currently open database.
Set database = CreateObject("actole.database") database.OpenEx ("")
Set contact = database.Contact contact.MoveFirst
```

```
'Each value in fieldarray must have a corresponding value in dataarray.
fieldarray(0) = 25
fieldarray(1) = 26
dataarray(0) = "Chris"
dataarray(1) = "13 East 54th Street"

'Start editing.
contact.Edit
'Set the data on the record.
contact.SetDataEx fieldarray, dataarray
'If there is not an error, update the record.
if contact.Error = FALSE then
  contact.Update
End If

Set contact = Nothing
database.Close
Set database = Nothing
```

SetDuplicateCriteria Method

Requires ACT! 5.0 or later

Description Sets field IDs for the duplicate checking criteria for the Contact table or Group table.

Object Contact object, Group object

Syntax *object*.**SetDuplicateCriteria** (*iFieldID1, iFieldID2, iFieldID3*)

Parameters *iFieldID1* A short integer (or pointer to a short integer in VISUAL C++) specifying the field ID for the first field for duplicate checking (find a match for data in this field).

iFieldID2 A short integer (or pointer to a short integer in VISUAL C++) specifying the field ID for the second field for duplicate checking (then find a match for data in this field).

iFieldID3 A short integer (or pointer to a short integer in VISUAL C++) specifying the field ID for the third field for duplicate checking (then find a match for data in this field).

See ACT! Databases for lists of field IDs and names (Field constants).

Return type Void

See also FindDuplicates Method, GetDuplicateCriteria Method

Example

```
Set objDatabase = CreateObject("ACTOLE.DATABASE")
'Open the database
objDatabase.Open dbName

'Assign the correct table object
Set objContact = objDatabase.Contact

If objDatabase.IsOpen Then
  'Get the current duplicate criteria
  objContact.GetDuplicateCriteria DupCriteria1, DupCriteria2, DupCriteria3
  List1.AddItem "Duplicate criteria is :  Match on   " & DupCriteria1 & "
    and then on  " & DupCriteria2 & "  and then on " & DupCriteria3

  'Set the duplicate criteria as match on User1 then Department and then on
    ' Title
  objContact.SetDuplicateCriteria CF_User1, CF_Department, CF_Title
```

```
    Set objContact = Nothing
    objDatabase.close
    End If

  Set objDatabase = Nothing
```

Sort Method

Description Executes a sort on up to three fields in a table. Only one sort may be active at once. An error is generated if any of the specified fields do not exist in the table.

Note: Call fields.IsSortable to determine if a field can be sorted.

Objects Activity object, Contact object, Group object, Members object, NoteHistory object

Also applies to the ListTable object and Sales object in ACT! 5.0 or later.

Syntax *object*.**Sort** *iField1ID, iDirection1, iField2ID, iDirection2, iField3ID, iDirection3*

Parameters *iField1ID* A short integer that specifies the field ID type for the field that specifies the first field in the sort order.

iDirection1 A short integer that specifies the sort direction for the first field. Specify a value of 0 for ascending order or 1 for descending order.

iField2ID A short integer that specifies the field ID for the field that specifies the second field in the sort order. Specify 0 to disable sorting on this field.

iDirection2 A short integer that specifies the sort direction for the second field. Specify 0 for ascending order or 1 for descending order.

iField3ID A short integer that specifies the field ID for the field that specifies the third field in the sort order. Specify 0 to disable sorting on this field.

iDirection3 A short integer that specifies the sort direction for the third field. Specify 0 for ascending order or 1 for descending order.

See ACT! Databases for lists of field IDs and names (Field constants).

Return type Void

See also Error Property, GetSort Method, IsBOF Property, LastError Property, RecordCount Property

IsSortable Property in Fields object

Example

```
'This example demonstrates how to perform a sort on the contact table.
dim objDatabase as object
Set objDatabase = CreateObject("ACTOLE.DATABASE")
objDatabase.Open dbName

'Set sort order for all 3 fields
objDatabase.CONTACT.Sort CF_Company, 1, CF_Title, 1, CF_Name, 1

'Set sort order for only the 1st field
objDatabase.CONTACT.Sort CF_Company, 1, 0, 1, 0, 1

objDatabase.Close
set objDatabase = Nothing
```

Update Method

Description Saves a new or edited record to the table. This method returns a string that contains the Unique ID of the record that has just been added or updated.

Objects Activity object, Contact object, Group object, NoteHistory object

Also applies to the ListTable object and Sales object in ACT! 5.0 or later.

Syntax *object*.**Update**

Return type String

Comments Make a call to Add or Edit before calling this method. This method commits a newly created or edited record to disk and removes the record lock. After updating, the current record position is determined by the alphabetical sort order of the field.

See also Add Method, Data Property, Edit Method, GoTo Method, MoveFirst Method, MoveLast Method, MoveNext Method, MovePrevious Method

Example

```
'This example demonstrates how to update a record in the contact table and
 get the UniqueID of the updated record.

Dim objDatabase as object
Dim UID as String * 12
Set objDatabase = CreateObject("ACTOLE.DATABASE")
objDatabase.Open dbName

'Set the database object to Edit mode.
objDatabase.Contact.Edit
    objDatabase.Contact.Data(CF_Company) = "Huffman Enterprises"
UID = objDatabase.Contact.Update

objDatabase.Close
Set objDatabase = Nothing
```

Chapter 3
Objects Derived from the Database Class

This chapter discusses the objects derived from the Database class and their associated methods and properties. Objects include:

Name	Name	Name
Activity object	ExceptionInfo object	NoteHistory object
Contact object	Fields object	PopupInfo object
Database object	Group object	Relations object
Email object	ListTable object	Sales object
ExceptionInfo object	Members object	Users object

Activity object

The Activity object contains scheduled activity information for the active database object. The following properties and methods apply only to the Activity object. See Common properties and methods for additional properties and methods that apply to this object.

Sample Code

The following sample uses the Activity object and adds an activity to a database.

```
Dim objDatabase As Object
Dim objactivity As Object
Dim objContact As Object
Dim strMsg As String
Dim strcontactID As String
dim strContactID2 as String
Dim stractivityID As String
Dim iduration As Integer
Dim dCurrentDate As Date

'This opens and logs on the database that is currently open in ACT!
'If ACT! is not open and logged on, this fails. Use the Open and
'ValidateUser methods to open a different database.
Set objDatabase = CreateObject("ACTOLE.DATABASE")
objDatabase.openex ""

If objDatabase.IsOpen = "" Then
    MsgBox "Failed to connect to ACT!.  Open ACT! and log on a database,
  then run the program again"
    Exit Sub
End If

Set objactivity = objDatabase.activity
Set objContact = objDatabase.contact
```

```
'Set field values for the contact record
strcontactID = objContact.Data(CF_UniqueID) 'get the contact unique ID
objContact.movenext 'move to another contact, assumes more than 1 contact
    'in the database
strcontactID2 = objContact.Data(CF_UniqueID) 'get the unique ID of another
    'contact
strMsg = "This is the regarding text for a 37 minute activity"
iduration = 37  'The duration of this activity will be 37 minutes
dCurrentDate = Now  'Sets dCurrentDate to the current date and time

objactivity.Add

'Sets the activity to public. Required or activity functions incorrectly
objactivity.Data AF_PublicPrivate, 1
objactivity.Data AF_AlarmStatus, 0  'Sets alarm to no alarm

'Start time, end time, and duration are required.
'Sets start time to dcurrentdate.  Despite the format of the data in
'the START_TIME field, when setting this value use a Date type variable.
objactivity.Data AF_StartTime, dCurrentDate
'Sets end date to dCurrentDate plus duration + 1 minute. End time must be
'start time plus duration + 1 minute.
objactivity.Data AF_EndTime, DateAdd("n", (iduration + 1), dCurrentDate)
objactivity.Data AF_Duration, iduration 'set the duration

objactivity.Data AF_Regarding, strMsg 'Sets the regarding text
objactivity.Data AF_Type, 0  'set type
objactivity.Data AF_Priority, 0 'set priority

'Update the activity and get the unique ID of the activity created
stractivityID = objactivity.Update

objactivity.goto (stractivityID) 'Go to the activity that was just created
'You must set firstscheduledwith or the activity will not function.
objactivity.firstscheduledwith (strcontactID)
'Set next scheduledwith if there are additional contacts with whom the
'activity should be scheduled
objactivity.nextscheduledwith (strcontactID2)

Set objactivity = Nothing
Set objContact = Nothing
objDatabase.Close
Set objDatabase = Nothing
```

Properties

Name	Parameter(s)	Parameter type(s)	Value type	Access
ExceptionInfo Property	None		Object/LPDISPATCH	Read Only
FirstScheduledWith Property	[szUniqueID]	String	String	Read/Write
IsOutlookActivity Property	None		Boolean	Read Only
NextScheduledWith Property	[szUniqueID]	String	String	Read/Write
RecurringChangeMode Property	[True|False]	Boolean	Boolean	Read/Write
RecurringType Property	None		Short Integer	Read Only

ExceptionInfo Property

Description Returns an ExceptionInfo object for a recurring activity.

Object Activity object

Syntax *object*.**ExceptionInfo**

Value type Object/LPDISPATCH, Read Only

FirstScheduledWith Property

Description Returns or sets the Unique ID of a contact for a single activity that is scheduled with multiple contacts. Use this property to add a contact to the list of contacts with whom an activity is scheduled. This is an important property as it makes ACT! display the activity in the Activities tab of the contact record.

Note: Do not call this property between the Add/Update or Edit/Update pairs.

Object Activity object

Syntax *object*.**FirstScheduledWith** [*szUniqueID*]

Parameters *szUniqueID* A string that specifies the Unique ID of a contact to add for a scheduled activity. Omit this optional parameter to get the contact record's Unique ID.

Value type String, Read/Write

Comments Use the FirstScheduledWith property to return or set the first contact for an activity, then use the NextScheduledWith property as many times as necessary to return or set each additional contact for the activity. When all contacts have been returned, a null string is returned.

See also NextScheduledWith Property

Example See the Activity object Sample Code.

IsOutlookActivity Property

Requires ACT! 5.0 or later

Description Returns True if the current activity is an Outlook activity and False if it is not an Outlook activity.

Object Activity object

Syntax *object*.**IsOutlookActivity**

Return type Boolean

Example

```
'The following code lists all the activities in the current database that
'are Outlook Activities.
Set objDatabase = CreateObject("ACTOLE.DATABASE")
objDatabase.Open dbName
Set objActivity = objDatabase.ACTIVITY

If objDatabase.IsOpen Then
    'Enumerate all of the records in the Activity table.
    objActivity.MoveFirst
    While Not objActivity.IsEOF
        'If the current activity record is an Outlook activity list it.
        If objActivity.IsOutlookActivity = True then
            List1.AddItem "Regarding: " & objActivity.Data(AF_Regarding)
            List1.AddItem "Start Time: " & objActivity.Data(AF_StartTime)
            List1.AddItem "Duration: " & objActivity.Data(AF_Duration)
        End if
        objActivity.MoveNext
    Wend
```

```
    Set objActivity = Nothing
    objDatabase.close
End If

Set objDatabase = Nothing
```

NextScheduledWith Property

Requires ACT! 4.0 or later

Description Returns or sets the Unique ID of a contact for a single activity that is scheduled with multiple contacts. Use this property to add a contact to the list of contacts with whom an activity is scheduled.

Note: Do not call this property between the Add/Update or Edit/Update pairs.

Object Activity object

Syntax *object*.**NextScheduledWith** [*szUniqueID*]

Parameters *szUniqueID* A string that specifies the Unique ID of a contact to add for a scheduled activity. Omit this optional parameter to get the contact record's Unique ID.

Value type String, Read/Write

Comments Use the FirstScheduledWith property to return or set the first contact for an activity, then use the NextScheduledWith property as many times as necessary to return or set each additional contact for the activity. When all contacts for the activity have been returned, a null string is returned.

See also FirstScheduledWith Property

Example See the Activity object Sample Code.

RecurringChangeMode Property

Description Gets and sets the change mode for a recurring activity. If True is set or returned, the change is to be applied for only the current instance of the recurring activity. If False is set or returned, the change is to be applied for all instances of the recurring activity.

Object Activity object

Syntax *object*.**RecurringChangeMode** *[True|False]*

Parameters *[True|False]* Specify True to set the change mode to the current instance of the recurring activity or False to set the change mode to all instances of the recurring activity. Omit this optional parameter to get the change mode for the recurring activity.

Value type Boolean, Read/Write

RecurringType Property

Description Returns the type of recurring activity. To get valid results, first verify that the activity is recurring by calling IsRecurring.

Object Activity object

Syntax *object*.**RecurringType**

Value type Short Integer, Read Only

Comments The following recurring type values are returned by this property:

Type	Description
0	Not recurring type
1	Daily recurring type
2	Weekly recurring type

Type	Description
3	Custom (days of the month) recurring type
4	Monthly recurring type

See also ClearRecurring Method, GetDaysOfMonthBits Method, GetDaysOfWeekBits Method, GetWeeksOfMonthBits Method, IsRecurring Method

Example

```
'This example demonstrates how to determine the type of recurring activity.

dim objDatabase as object
Set objDatabase = CreateObject("ACTOLE.DATABASE")
objDatabase.Open dbN
objDatabase.ACTIVITY.MoveFirst

if objDatabase.ACTIVITY.IsRecurring then
    recurtype = objDatabase.ACTIVITY.RecurringType
    select case recurtype
        case recurring_none
        case recurring_days
            'Daily recurring activity
        case recurring_weekdays
            'Weekly recurring activity
        case recurring_daysofmonth
            'Custom recurring activity
        case recurring_daysandweeksofmonth
            'Monthly recurring activity
    end select
end if

objDatabase.Close
set objDatabase = Nothing
```

Methods

Name	Parameter(s)	Parameter type(s)	Return type
Clear Method	None		Boolean
ClearClearedFilter Method	None		Void
ClearContactScope Method	None		Void
ClearDateScope Method	None		Void
ClearGroupScope Method	None		Void
ClearPriorityFilter Method	None		Void
ClearRecurring Method	None		Void
ClearTimedFilter Method	None		Void
ClearTimelessFilter Method	None		Void
ClearTypeFilter Method	None		Void
ClearUnclearedFilter Method	None		Void
GetDaysOfMonthBits Method	None		Long Integer
GetDaysOfWeekBits Method	None		Long Integer

Name	Parameter(s)	Parameter type(s)	Return type
GetRecurringFrequency Method	None		Short Integer
GetRecurringUntilDate Method	None		Date/Variant
GetWeeksOfMonthBits Method	None		Long Integer
HasAlarm Method	None		Boolean
HasDetails Method	None		Boolean
IsClear Method	None		Boolean
IsRecurring Method	None		Boolean
IsTimeless Method	None		Boolean
SetClearedFilter Method	None		Void
SetContactScope Method	szUniqueID	String	Void
SetDateScope Method	startDate, endDate	Date, Date	Void
SetGroupScope Method	szGroupUniqueID	String	Void
SetPriorityFilter Method	iPriority	Short Integer	Void
SetRecurringDays Method	lFrequency, untilDate	Long Integer, Date	Void
SetRecurringDaysAndWeeksofMonth Method	lFrequency DayBits lWeekBits untilDate	Long Integer Long Integer Long Integer Date	Void
SetRecurringWeekDays Method	lFrequency lDayBits untilDate	Long Integer Long Integer Date	Void
SetTimedFilter Method	None		Void
SetTimeless Method	True|False	Boolean	Void
SetTimelessFilter Method	None		Void
SetTypeFilter Method	iType	Short Integer	Void
SetUnclearedFilter Method	None		Void
Unclear Method	None		Boolean

Clear Method

Description Sets the status of the current activity to cleared.

Object Activity object

Syntax *object*.**Clear**

Return type Boolean

See also IsClear Method

ClearClearedFilter Method

Description Clears an existing cleared filter. Resets the current record position to the first record and rebuilds the list of activities, without applying the cleared filter.

Object Activity object

Syntax *object*.**ClearClearedFilter**

Return type Void

See also SetClearedFilter Method

ClearContactScope Method

Description Clears any existing contact scoping. Resets the current record position to the first record and rebuilds the list of activities, without applying contact scoping.

Object Activity object

Syntax *object*.**ClearContactScope**

Return type Void

See also SetContactScope Method

ClearDateScope Method

Description Clears any existing date scoping. Resets the current record position to the first record and rebuilds the list of activities, without applying date scoping.

Object Activity object

Syntax *object*.**ClearDateScope**

Return type Void

See also SetDateScope Method

ClearGroupScope Method

Description Clears any existing group scoping. Resets the current record position to the first record and rebuilds the list of activities, without applying group scoping.

Object Activity object

Syntax *object*.**ClearGroupScope**

Return type Void

See also SetGroupScope Method

ClearPriorityFilter Method

Description Clears all existing priority filters. Resets the current record position to the first record and rebuilds the list of activities, without applying any priority filter.

Object Activity object

Syntax *object*.**ClearPriorityFilter**

Return type Void

See also SetPriorityFilter Method

ClearRecurring Method

Description Changes the recurring status of an activity status to non-recurring, making a recurring activity non-recurring.

Object Activity object

Syntax *object*.**ClearRecurring**

Return type Void

See also RecurringType Property, SetRecurringDays Method, SetRecurringDaysAndWeeksofMonth Method, SetRecurringWeekDays Method

ClearTimedFilter Method

Description Clears an existing timed filter and rebuilds a list of records that includes any timed activities.

Object Activity object

Syntax *object*.**ClearTimedFilter**

Return type Void

See also SetTimedFilter Method

ClearTimelessFilter Method

Description Clears an existing timeless filter and rebuilds a list of records that includes all timeless activities.

Object Activity object

Syntax *object*.**ClearTimelessFilter**

Return type Void

See also SetTimelessFilter Method

ClearTypeFilter Method

Description Clears all existing activity type filters and rebuilds a list of records that includes all activities, including activities with types that were previously filtered out.

Object Activity object

Syntax *object*.**ClearTypeFilter**

Return type Void

See also SetTypeFilter Method

ClearUnclearedFilter Method

Description Clears an existing uncleared filter and rebuilds a list of records that includes all activities including the uncleared activities that were previously filtered out.

Object Activity object

Syntax *object*.**ClearUnclearedFilter**

Return type Void

See also SetUnclearedFilter Method

GetDaysOfMonthBits Method

Description Returns a long integer that specifies the days of a month that are set for a recurring activity. To get valid results, call the RecurringType property before calling this method. The returned value represents a value equivalent to 2^ (Day of the month -1).

Object Activity object

Syntax *object*.**GetDaysOfMonthBits**

Return type Long Integer

Comments The following values for days of a month are returned by this method:

Value	Day of month	Value	Day of month	Value	Day of month
1	1	2048	12	2097152	22
2	2	4096	13	4194304	23
4	3	8192	14	8388608	24
8	4	16384	15	16777216	25
16	5	32768	16	33554432	26
32	6	65536	17	67108864	27
64	7	131072	18	134217728	28
128	8	262144	19	268435456	29

Value	Day of month	Value	Day of month	Value	Day of month
256	9	524288	20	536870912	30
512	10	1048576	21	1073741824	31
1024	11				

See also IsRecurring Method, RecurringType Property

Example

```
Dim objDatabase As Object
Set objDatabase = CreateObject("ACTOLE.DATABASE")

objDatabase.Open dbName
List1.Clear
objDatabase.ACTIVITY.MoveFirst

If objDatabase.ACTIVITY.IsRecurring Then
    recurtype = objDatabase.ACTIVITY.RecurringType

    If recurtype = recurring_daysofmonth Then
    Dim daysOfMonth as long
    daysOfMonth = objDatabase.ACTIVITY.GetDaysOfMonthBits()

    'Get the day of the month
        Dim i As Integer
        Dim dayvalue as long
        Dim outString as String
        For i = 0 To 30
            dayvalue = 2 ^ i

            If daysOfMonth And dayvalue Then
                outString = outString & (i + 1) & " "
            End If
        Next i
        'OutString now has the value of the day of the month
        '(number between 1 and 31)
        List1.AddItem "Days of the Month: " & outString
    End If
End If
```

GetDaysOfWeekBits Method

Description Returns a long integer that specifies the days of the week that are set for a recurring activity. To get valid results, first call the RecurringType property and check that the activity is a recurring_weekdays or a recurring_daysandweeksofmonth type.

Object Activity object

Syntax *object*.**GetDaysOfWeekBits**

Return type Long Integer

Comments The returned long integer value must be ANDed with specific defines representing the week bit set.

The following values for days of a week are returned by this method:

Value	Day of week	Value	Day of week
1	Sunday	16	Thursday
2	Monday	32	Friday

Value	Day of week	Value	Day of week
4	Tuesday	64	Saturday
8	Wednesday		

See also GetDaysOfMonthBits Method, GetWeeksOfMonthBits Method, RecurringType Property

Example

```
dim objDatabase as object
Set objDatabase = CreateObject("ACTOLE.DATABASE")
objDatabase.Open dbName

'Example demonstrates how to parse values returned from GetDaysOfWeekBits
objDatabase.ACTIVITY.MoveFirst
if objDatabase.ACTIVITY.IsRecurring then
    recurtype = objDatabase.ACTIVITY.RecurringType

    if recurtype = recurring_weekdays or recurtype =
recurring_daysandweeksofmonth then
        dim dayBits as long
        dayBits = objDatabase.ACTIVITY.GetDaysOfWeekBits
        'Parse the day bits
        If dayBits And day_sunday Then
            'Sunday is set
        End If

        If dayBits And day_monday Then
            'Monday is set
        End If
    End if
End if

objDatabase.Close
set objDatabase = Nothing
```

GetRecurringFrequency Method

Description Returns a short integer that specifies the frequency of a recurring activity.

Object Activity object

Syntax *object*.**GetRecurringFrequency**

Return type Short Integer

Comments Verify that the current activity is recurring by calling IsRecurring and determine the type of recurring activity by calling RecurringType to know the time period. For example, if RecurringType is 2 and GetRecurringFrequency is 1, the activity recurs every week.

See also IsRecurring Method, RecurringType Property

Example

```
dim objDatabase as object
Set objDatabase = CreateObject("ACTOLE.DATABASE")
objDatabase.Open dbName

'Example demonstrating how to determine the type of recurring activity
objDatabase.ACTIVITY.MoveFirst

if objDatabase.ACTIVITY.IsRecurring then
    dim outString as string
    outString = "Recurring every: "
```

```
    recurtype = objDatabase.ACTIVITY.RecurringType

    select case recurtype
        case recurring_none
        case recurring_days
            'Daily recurring activity
            outString = outString & activity.GetRecurringFrequency()
                & " Days"
        case recurring_weekdays
            'Weekly recurring activity
            outString = outString & activity.GetRecurringFrequency()
                & " Weeks"
        case recurring_daysofmonth
            'Custom recurring activity
            outString = outString & activity.GetRecurringFrequency()
                & " Months"
        case recurring_daysandweeksofmonth
            'Monthly recurring activity
        outString = outString & activity.GetRecurringFrequency() & " Months"
    end select
end if

objDatabase.Close
set objDatabase = Nothing
```

GetRecurringUntilDate Method

Description Returns the date on which the recurring activity will stop recurring. In Visual Basic you can Typecast it as a string.

Object Activity object

Syntax *object*.**GetRecurringUntilDate**

Return type Date/Variant

GetWeeksOfMonthBits Method

Description Returns a long integer that specifies which weeks of the month are set for a recurring activity. To get valid results, call RecurringType and check that this activity is a recurring_daysandweeksofmonth type.

Object Activity object

Syntax *object*.**GetWeeksOfMonthBits**

Return type Long Integer

Comments The long integer value returned must be ANDed with specific defines representing which week bit is set.

The following values for weeks of a month are returned by this method:

Value	Week of month	Value	Week of month
1	one	8	four
2	two	16	last
4	three		

See also GetDaysOfWeekBits Method, IsRecurring Method, RecurringType Property

Example

```
'Demonstrates how to parse values returned from GetWeeksOfMonthBits
dim objDatabase as object
Set objDatabase = CreateObject("ACTOLE.DATABASE")
objDatabase.Open dbName

objDatabase.ACTIVITY.MoveFirst
if objDatabase.ACTIVITY.IsRecurring then
    recurtype = objDatabase.ACTIVITY.RecurringType

    if recurtype = recurring_daysandweeksofmonth then
        dim weekBits as long
        weekBits = objDatabase.ACTIVITY.GetWeeksOfMonthBits
        'Parse the weekbits
        If weeksBits And week_one Then
            'Week one is set
        End If

        If weekBits And week_two Then
            'Week two is set
        End If
    End if
End if

objDatabase.Close
set objDatabase = Nothing
```

HasAlarm Method

Description Returns True if an alarm has been set for the current activity or False if an alarm has not been set. Call this method to determine if an activity currently has an alarm set.

Object Activity object

Syntax *object*.**HasAlarm**

Return type Boolean

Comments This method cannot be used to set an alarm for an activity.

HasDetails Method

Requires ACT! 5.0 or later

Description Determines if the current activity has details associated with it. ReturnsTrue if the current activity includes details or False if the activity does not include details.

Object Activity object

Syntax *object*.**HasDetails**

Return type Boolean

Example

```
'Lists all the activities with details in the current database.
    Set objDatabase = CreateObject("ACTOLE.DATABASE")
    objDatabase.Open dbName
    Set objActivity = objDatabase.Activity

    If objDatabase.IsOpen Then
        'Enumerate all of the records in the Activity table.
        objActivity.MoveFirst
```

```
        While Not objActivity.IsEOF
            List1.AddItem "Regarding: " & objActivity.Data(AF_Regarding)
            List1.AddItem "Start Time: " & objActivity.Data(AF_StartTime)
            List1.AddItem "Duration: " & objActivity.Data(AF_Duration)
            If objActivity.HasDetails = True Then
                List1.AddItem "ACTIVITY Details: " &
                objActivity.Data(AF_Details)
            End If
            objActivity.MoveNext
        Wend
        Set objActivity = Nothing
        objDatabase.close
    End If
Set objDatabase = Nothing
```

IsClear Method

Description Returns True if the current activity has a status of cleared or False if the status is not cleared.

Object Activity object

Syntax *object*.**IsClear**

Return type Boolean

See also Clear Method, Unclear Method

IsRecurring Method

Description Returns True if the current activity is recurring or False if it occurs only once. Call this method before obtaining any type of recurring information.

Object Activity object

Syntax *object*.**IsRecurring**

Return type Boolean

See also ClearRecurring Method, GetDaysOfMonthBits Method, GetDaysOfWeekBits Method, GetWeeksOfMonthBits Method, IsRecurring Method

Example See RecurringType Property

IsTimeless Method

Description Returns True if the status of the current activity is timeless or False if the activity has a starting time of day.

Object Activity object

Syntax *object*.**IsTimeless**

Return type Boolean

See also SetTimeless Method

SetClearedFilter Method

Description Narrows the current set of activity records by filtering out all cleared activities from the current set of records. The list of records is rebuilt and the current record position returns to the first record.

Object Activity object

Syntax *object*.**SetClearedFilter**

Return type Void

See also ClearClearedFilter Method

SetContactScope Method

Description Narrows the current set of activity records to those for the contact with the specified Unique ID. You can apply this scoping method along with other scoping methods and filters.

Object Activity object

Syntax *object*.**SetContactScope** *szUniqueID*

Parameters *szUniqueID* A string that specifies the Unique ID of a contact record.

Return type Void

See also ClearContactScope Method

Example

```
dim objDatabase as object
Set objDatabase = CreateObject("ACTOLE.DATABASE")
objDatabase.Open dbName

'Get a contact record Unique ID
dim uniqueID as string
objDatabase.CONTACT.MoveFirst
uniqueID = objDatabase.CONTACT.DATA(CF_UniqueID)

'Apply contact scoping to the current list of activities
objDatabase.ACTIVITY.MoveFirst
objDatabase.ACTIVITY.SetContactScope(uniqueID)

objDatabase.Close
set objDatabase = Nothing
```

SetDateScope Method

Description Narrows the current set of activity records to those within the specified start date and end date range. You can apply this scoping method along with other scoping methods and filters.

Object Activity object

Syntax *object*.**SetDateScope** *startDate*, *endDate*

Parameters *startDate* A date value that specifies the beginning date of the activity date range, formatted in Windows Regional Settings Short Date style.

endDate A date value that specifies the ending date of the activity date range, formatted in Windows Regional Settings Short Date style.

Return type Void

See also ClearDateScope Method

Example

```
dim objDatabase as object
Set objDatabase = CreateObject("ACTOLE.DATABASE")
objDatabase.Open dbName

'Locate all activity records that fall between these dates
objDatabase.ACTIVITY.MoveFirst
objDatabase.ACTIVITY.SetDateScope DateValue("01/15/97"),
 DateValue("03/01/97")

objDatabase.Close
set objDatabase = Nothing
```

SetGroupScope Method

Description Narrows the current set of activity records to those for the group with the specified Unique ID. You can apply this scoping method along with other scoping methods and filters.

Object Activity object

Syntax *object*.**SetGroupScope** *szGroupUniqueID*

Parameters *szGroupUniqueID*A string that specifies the Unique ID of a group record.

Return type Void

See also ClearGroupScope Method

SetPriorityFilter Method

Description Narrows the current set of activity records by filtering out all activity records with one or more specified priorities from the current set of records. The list of records is rebuilt, and the record position returns to the first record.

Object Activity object

Syntax *object*.**SetPriorityFilter** *iPriority*

Parameters *iPriority* A short integer that specifies the activity record priority type to filter out. Specify 0 to filter out high priority activities, 1 for medium priority activities, or 2 for low priority activities.

Return type Void

Comments To filter out activities with a second priority type, call this method again with another parameter value. You may want to filter out both low and medium priority activities, for example, which means you will be left with a list of only high priority activities.

See also ClearPriorityFilter Method

SetRecurringDays Method

Description Sets the current activity record to recurring with a recurring type of daily and specifies the date when the activity will end.

Note: Do not call this method between Add/Update and Edit/Update pairs.

Object Activity object

Syntax *object*.**SetRecurringDays** *lFrequency*, *untilDate*

Parameters *lFrequency* A long integer that specifies every x days of the month, in the range of 1 to 31, depending on the number of days in the month.

untilDate A date value that specifies the date when the recurring activity will end, formatted in Windows Regional Settings Short Date style. This value must be greater than the start date of the current activity.

Return type Void

See also RecurringType Property, SetRecurringDaysAndWeeksofMonth Method, SetRecurringWeekDays Method

Example

```
dim objDatabase as object
Set objDatabase = CreateObject("ACTOLE.DATABASE")
objDatabase.Open dbName
objDatabase.ACTIVITY.MoveFirst

'Alter the activity so that it happens every 2 days until 3/26/99
objDatabase.ACTIVITY.SetRecurringDays 2, DateValue("03/26/99")
```

```
objDatabase.Close
set objDatabase = Nothing
```

SetRecurringDaysAndWeeksofMonth Method

Description Sets the current activity record to recurring with a recurring type of monthly (days and weeks of a month) and specifies the date when the activity will end. Values for both days of the week bits and weeks of the month bits must be passed to this method.

Note: Do not call this method between Add/Update and Edit/Update pairs.

Object Activity object

Syntax *object*.**SetRecurringDaysAndWeeksofMonth** *lFrequency, lDayBits, lWeekBits, untilDate*

Parameters *lFrequency* A long integer that species every x months, in a range of 1 to 60.

lDayBits A long integer that specifies the days of the week. This value is created by ORing together values that represent the days of the week.

The following table lists the values for each day of a week:

Value	Day of week	Value	Day of week
1	Sunday	16	Thursday
2	Monday	32	Friday
4	Tuesday	64	Saturday
8	Wednesday		

lWeekBits A long integer that specifies the weeks of the month. This value is created by ORing together values that represent the weeks of the month.

The following table lists the values for each week of a month:

Value	Week of month	Value	Week of month
1	one	8	four
2	two	16	last
4	three		

untilDate A date value that specifies the date when the recurring activity will end, formatted in Windows Regional Settings Short Date style. This value must be greater than the start date of the current activity.

Return type Void

See also RecurringType Property, SetRecurringDays Method, SetRecurringWeekDays Method

Example

```
dim objDatabase as object
Set objDatabase = CreateObject("ACTOLE.DATABASE")
objDatabase.Open dbName

objDatabase.ACTIVITY.MoveFirst

'Alter the activity so that it happens every one month on Sunday
'and on Thursday of the second and fourth weeks of the month up until
'the date 3/26/99
dim dayBits as long
dim weekBits as long
dayBits = day_sunday Or day_Thursday
```

```
weekBits = week_two Or week_four
objDatabase.ACTIVITY.SetRecurringDaysAndWeeksofMonth 1 dayBits, weekbits,
    DateValue("03/26/99")

objDatabase.Close
set objDatabase = Nothing
```

SetRecurringWeekDays Method

Description Sets the current activity record to recurring with a recurring type of weekly and specifies when the activity will end.

Note: Do not call this method between Add/Update and Edit/Update pairs.

Object Activity object

Syntax *object.***SetRecurringWeekDays** *lFrequency, lDayBits, untilDate*

Parameters *lFrequency* A long integer representing every x weeks, in the range of 1 to 52.

lDayBits A long integer representing the days of the week. This value is created by ORing together values which represent the days of the week.

The following table lists the values for each day of a week:

Value	Day of week	Value	Day of week
1	Sunday	16	Thursday
2	Monday	32	Friday
4	Tuesday	64	Saturday
8	Wednesday		

untilDate A date value that specifies the date when the recurring activity will end, formatted in Windows Regional Settings Short Date style. This value should be greater than the start date of the current activity.

Return type Void

See also RecurringType Property, SetRecurringDays Method, SetRecurringDaysAndWeeksofMonth Method

Example

```
dim objDatabase as object
Set objDatabase = CreateObject("ACTOLE.DATABASE")
objDatabase.Open dbName

objDatabase.ACTIVITY.MoveFirst

'Alter an activity that happens every 2 weeks on Sunday and Thursday and
'continues until 3/26/03
dim dayBits as long
dayBits = day_sunday Or day_thursday
objDatabase.ACTIVITY.SetRecurringWeekDays 2, dayBits,
 DateValue("03/26/03")

objDatabase.Close
set objDatabase = Nothing
```

SetTimedFilter Method

Description Narrows the current set of activity records by filtering out all timed activity records from the current set of records. The list of records is rebuilt, and the record position returns to the first record.

Object Activity object

Syntax *object*.**SetTimedFilter**

Return type Void

See also ClearTimedFilter Method

SetTimeless Method

Description Sets the current activity record's activity status.

Object Activity object

Syntax *object*.**SetTimeless** *True|False*

Parameters *True|False* Specify True to set the activity to timeless or False to set the activity to not timeless.

Return type Void

See also IsTimeless Method

SetTimelessFilter Method

Description Narrows the current set of activity records by filtering out all timeless activities from the current set of records. The list of records is rebuilt, and the record position returns to the first record.

Object Activity object

Syntax *object*.**SetTimelessFilter**

Return type Void

See also ClearTimelessFilter Method

SetTypeFilter Method

Description Narrows the current set of activity records by filtering out all activity records with the specified type from the current set of records. The list of records is rebuilt, and the record position returns to the first record.

Object Activity object

Syntax *object*.**SetTypeFilter** *iType*

Parameters *iType* A short integer that specifies the type of activity records to filter out. Specify 0 to filter out Call type activities, 1 for Meeting type activities, or 2 for To-do type activities.

Return type Void

Comments To filter out activities with a second activity type, call this method again with another parameter value. You may want to filter out both Call and Meeting type activities, for example, which means you will be left with a list of only To-do type activities.

See also ClearTypeFilter Method

Example

```
dim objDatabase as object
Set objDatabase = CreateObject("ACTOLE.DATABASE")
objDatabase.Open dbName

objDatabase.ACTIVITY.MoveFirst

'Filter out all calls from the Activity list
objDatabase.ACTIVITY.SetTypeFilter(activitytype_call) 'defined in
 ACTFIELD.BAS
```

```
objDatabase.Close
set objDatabase = Nothing
```

SetUnclearedFilter Method

Description Narrows the current set of activity records by filtering out all uncleared activity records from the current set of records. The list of records is rebuilt, and the record position returns to the first record.

Object Activity object

Syntax *object*.**SetUnclearedFilter**

Return type Void

See also ClearUnclearedFilter Method

Unclear Method

Description Resets the cleared status of the current activity record to uncleared.

Object Activity object

Syntax *object*.**Unclear**

Return type Boolean

See also Clear Method

Contact object

The Contact object contains information about the ACT! contacts. The following methods apply only to the Contact object. See Common properties and methods for additional properties and methods that apply to this object.

Methods

Name	Parameter(s)	Parameter type(s)	Return type
LoadLookUpQuery Method	szFileName	String	Void
LookupMyRecord Method	None		Void
SetAsMyRecord Method	szUserName	String	Void

LoadLookUpQuery Method

Requires ACT! 3.0.6 or later

Description Loads and runs a file that was created using the SaveCurrentLookup method of the Application object.

Note: This method cannot use queries created in the ACT! application.

Object Contact object

Syntax *object*.**LoadLookUpQuery** *szFileName*

Parameters *szFileName* A string that specifies the name and path of the Query file.

Return type Void

See also SaveCurrentLookup Method of the Application object

LookupMyRecord Method

Requires ACT! 5.0 or later

Description Looks up the My Record in the open database. After executing this command, the Current lookup contains one contact, the My Record.

Object Contact object

Syntax *object*.**LookupMyRecord**

Return type Void

Example

```
'The following code looks up the My Record and Lists the Name, Title, and
'Company information.
Set objDatabase = CreateObject("ACTOLE.DATABASE")
'Opens the currently open database in ACT!
objDatabase.OpenEx ""

Set objContact = objDatabase.CONTACT
If objDatabase.IsOpen Then
    'Lookup My Record and list the My Record information.
    objContact.LookupMyRecord
    List1.AddItem " My Record"
    List1.AddItem "NAME: " & objContact.Data(CF_Name)
    List1.AddItem "TITLE: " & objContact.Data(CF_Title)
    List1.AddItem "COMPANY: " & objContact.Data(CF_Company)
    Set objContact = Nothing
    objDatabase.close
    List1.AddItem "Database Closed"
End If

Set objDatabase = Nothing
```

SetAsMyRecord Method

Requires ACT! 4.0 or later

Description Makes the current contact record the My Record of the specified user. To use this method, the current user of the open database and the specified user must have an Administrator security level. The specified user must not have an existing My Record in the open database. A contact record that was previously assigned to a user cannot be assigned to other users.

Object Contact object

Syntax *object*.**SetAsMyRecord** *szUserName*

Parameters *szUserName* A string that specifies the user name.

Return type Void

Example

```
'This example sets a contact record as the My Record for a user.
dim objDatabase as object
dim objContact as Object
dim objUsers as Object

Set objDatabase = CreateObject("ACTOLE.DATABASE")
objDatabase.Open("C:\My Documents\ACT\Database\ACT5demo.dbf")
Set objUsers = objDatabase.Users

'Add a user.
if objUsers.AddUser("Simon Lazarus", "password", 0) = True then
```

```
    'Get the Contact object.
    Set objContact = objDatabase.CONTACT
    objContact.Add
    objContact.Data (CF_Name, "Simon Lazarus")
    objContact.Data (CF_Company, "Aussie Meats")
    'Save the contact and get the Unique ID.
    objContact.Update
    'Mark the record as the My Record for user Simon Lazarus.
    objContact.SetAsMyRecord("Simon Lazarus")
Endif

objDatabase.Close
set objDatabase = Nothing
```

Database object

The Database object contains the ACT! database information and objects. The following properties and methods apply only to the Database object. See Common properties and methods for additional properties and methods that apply to this object.

Properties

Name	Parameter(s)	Parameter type(s)	Value type	Access
ActiveUserCount Property	None		Short Integer	Read Only
Activity Property	None		Object/LPDISPATCH	Read Only
ActVersion Property	None		String	
Contact Property	None		Object/LPDISPATCH	Read Only
CurrentUser Property	None		String	Read Only
DatabaseVersion Property	None		String	
Email Property	None		Object/LPDISPATCH	Read Only
Group Property	None		Object/LPDISPATCH	Read Only
IsInBatchMode Property	None		Boolean	Read Only
IsLocked Property	None		Boolean	Read Only
IsMultiUser Property	None		Boolean	Read Only
IsOpen Property	None		Boolean	Read Only
IsOpening Property	None		Boolean	Read Only
LogTransactions Property	[True|False]	Boolean	Boolean	Read/Write
Name Property	None		String	Read Only
NoteHistory Property	None		Object/LPDISPATCH	Read Only
PhoneFormatting Property	[iValue]	Short Integer	Short Integer	Read/Write
Relations Property	None		Object/LPDISPATCH	Read Only
TableCount Property	None		Short Integer	
Users Property	None		Object/LPDISPATCH	Read Only
Version Property	None		String	Read Only

ActiveUserCount Property

Description Returns the total number of local and remote users logged on to the current database, including the OLE automation client. This number changes as users log on and log off the database.

Object Database object

Syntax *object.***ActiveUserCount**

Value type Short Integer, Read Only

See also IsMultiUser Property, IsOpen Property

Count Property in Users object

Activity Property

Description Returns an Activity object. The Database object returns the same instance of the Activity object on each subsequent call.

Object Database object

Syntax *object.***Activity**

Value type Object/LPDISPATCH, Read Only

See also Activity object

Example This example demonstrates how to retrieve an Activity object.

```
dim objDatabase as object
Set objDatabase = CreateObject("ACTOLE.DATABASE")
objDatabase.Open dbName

'Obtain the Activity object
dim activity as object
set activity = objDatabase.ACTIVITY
set activity = Nothing

objDatabase.Close
set objDatabase = Nothing
```

ActVersion Property

Requires ACT! 3.0.6 or later

Description Returns a string that contains the version of ACT! used by the ACT! OLE Database object. An example of a returned string is 3.0.6.123, where 3.0.6.123 is the version of ACT!

Object Database object

Syntax *object.***ActVersion**

Value type String

See also Version Property

Contact Property

Description Returns a Contact object. The Database object returns the same instance of the Contact object on each subsequent call.

Object Database object

Syntax *object.***Contact**

Value type Object/LPDISPATCH, Read Only

See also Contact object

CurrentUser Property

Description Returns the name of the current user of the database. This property returns the logon name of the current user, not the contact name for that user.

Object Database object

Syntax *object*.**CurrentUser**

Value type String, Read Only

See also IsMultiUser Property, IsOpen Property

Count Property in Users object

DatabaseVersion Property

Requires ACT! 4.0 or later

Description Returns a string that contains the version of the current database, which is 3.0 or 4.0 for the ACT! 4.0 application or 5.0 for the ACT! 5.0 application.

Note: In ACT! 5.0, ACT! 3.0 and ACT! 4.0 format databases must be converted to ACT! 5.0 format.

Object Database object

Syntax *object*.**DatabaseVersion**

Value type String

Example

```
'This example checks for the OLE Database object version, then checksif the
 database is an ACT! 3.0 or ACT! 4.0 format database.
If (Val(objDatabase.Version) > 3) Then
   If (Val(objDatabase.DatabaseVersion) > 3) Then
   'Ticker Symbol field is available only in ACT! 4.0 format database.
   objDatabase.CONTACT.Data CF_TickerSymbol, "TS"
   End If
End If
```

Email Property

Description Returns an E-mail object. The Database object returns the same instance of the Email object on each subsequent call.

Object Database object

Syntax *object*.**Email**

Value type Object/LPDISPATCH, Read Only

See also Email object

Group Property

Description Returns a Group object. The Database object returns the same instance of the Group object on each subsequent call.

Object Database object

Syntax *object*.**Group**

Value type Object/LPDISPATCH, Read Only

See also Group object

IsInBatchMode Property

Requires ACT! 4.0.2 or later

Description Returns True if the current database is in batch mode for insertion of records or False if it is not in batch mode. Call this property immediately before inserting records into a database in batch mode.

Object Database object

Syntax *object*.**IsInBatchMode**

Value type Boolean, Read Only

See also BeginBatchInsert Method, BeginBatchUpdate Method, EndBatchInsert Method, EndBatchUpdate Method, IsLocked Property, Lock Method, Unlock Method

Example See BeginBatchInsert

IsLocked Property

Requires ACT! 4.0.2 or later

Description Returns True if the current database is locked or False if it is not locked. Call this property immediately before turning on batch mode for record insertion into a database.

Object Database object

Syntax *object*.**IsLocked**

Value type Boolean, Read Only

See also BeginBatchInsert Method, BeginBatchUpdate Method, EndBatchInsert Method, EndBatchUpdate Method, IsInBatchMode Property, Lock Method, Unlock Method

Example See BeginBatchInsert

IsMultiUser Property

Description Returns True if the open database is a multiuser database (or a password-protected single-user database) or False if it is a single-user database.

Object Database object

Syntax *object*.**IsMultiUser**

Value type Boolean, Read Only

Comments Multiuser databases must have two or more users who have logged on to the database and have generated a My Record contact record. Adding an ACT! user who hasn't logged in as that user at least once does not increase the user count.

See also IsOpen Property, IsOpening Property, ValidateUser Method

Count Property in Users object

Example See ValidateUser

IsOpen Property

Description Returns the open status of the database. ReturnsTrue after Open has been successfully called for a single-user database or afterValidateUser has been successfully called for a multiuser database. False is returned if Open has been called successfully for a multiuser database.

Object Database object

Syntax *object*.**IsOpen**

Value type Boolean, Read Only

See also IsMultiUser Property, IsOpening Property, ValidateUser Method

Example See the Activity object Sample Code.

IsOpening Property

Description Returns the current open status of a multiuser database. Returns True after Open has been successfully called for a multiuser database and prior to calling ValidateUser. Returns False after ValidateUser has been successfully completed.

Object Database object

Syntax *object*.**IsOpening**

Value type Boolean, Read Only

See also IsMultiUser Property, IsOpen Property, ValidateUser Method

Count Property in Users object

LogTransactions Property

Description Gets and sets the status of automatic transaction logging. This property returns or sets True if transaction log records will be generated when contact data is added, modified, or deleted or returns False if the transaction log records are not generated by changes to contact data. The default value is False.

Object Database object

Syntax *object*.**LogTransactions** *[True|False]*

Parameters *[True|False]* Specify True to turn on automatic transaction logging, or False to turn it off. Omit this optional parameter to get the status of automatic transaction logging.

Value type Boolean, Read/Write

Name Property

Description Returns a string containing the name of the currently open database, for example "ACT5DEMO.DBF".

Object Database object

Syntax *object*.**Name**

Value type String, Read Only

See also IsOpen Property, Open Method

NoteHistory Property

Description Returns a NoteHistory object. The Database object returns the same instance of the NoteHistory object on each subsequent call.

Object Database object

Syntax *object*.**NoteHistory**

Value type Object/LPDISPATCH, Read Only

PhoneFormatting Property

Description Gets and sets the format for phone numbers that are returned by the Data method.

Note: When setting data into a phone field, you can only pass aTAPI canonical-formatted phone number.

Object Database object

Syntax *object*.**PhoneFormatting** [*iValue*]

Parameters *iValue* A short integer that specifies the phone number format, in a range from 1 to 3. Omit this optional parameter to get the phone number format.

Value type Short Integer, Read/Write

Comments This property returns or sets one of the following values:

Value	Description
1	All phone numbers are returned in TAPI canonical format. This string may be passed to any TAPI components. This is the default setting.
2	All phone numbers are returned in ACT! 5.0 displayable format, which may not work with an automatic dialer.
3	All phone numbers are returned in ACT! 5.0 displayable format, starting with a country code enclosed in brackets [ ]. The default country code is the My Record phone number country code, which is [1] for the United States and Canada. This format may not work with an automatic dialer.

See also IsOpen Property, Open Method

Relations Property

Requires ACT! 3.0.6 or later

Description Returns the Relations object. The Database object returns the same instance of the Relations object on each subsequent call.

Object Database object

Syntax *object*.**Relations**

Value type Object/LPDISPATCH, Read Only

TableCount Property

Requires ACT! 3.0.6 or later

Description Returns a short integer that contains the total number of ACT! database tables. The value "5" is returned in ACT! 3.0.6 and 4.0. The value "7" is returned in ACT! 5.0.

Object Database object

Syntax *object*.**TableCount**

Value type Short Integer

Users Property

Description Returns a Users object. The Database object returns the same instance of the Users object on each subsequent call.

Object Database object

Syntax *object*.**Users**

Value type Object/LPDISPATCH, Read Only

Version Property

Requires ACT! 3.0.6 or later

Description Returns a string that contains the version of the ACT! OLE Database object. An example of a returned string is 5.0.0.175, where 5.0.0 is the version of ACT! (ACT! 5.0) and 175 is the number of the build. Use this property to verify the version to determine if new properties and methods in the OLE Database object can be used.

Object Database object

Syntax *object*.**Version**

Value type String, Read Only

See also ActVersion Property

Methods

Name	Parameter(s)	Parameter type(s)	Return type
BeginBatchInsert Method	iBatchFlag	Short Integer	Boolean
BeginBatchUpdate Method	iBatchFlag	Short Integer	Boolean
Close Method	None		Void
EndBatchInsert Method	None		Boolean
EndBatchUpdate Method	None		Boolean
GetDatabasePath Method	None		String
GetTableId Method	iIndex	Short Integer	Short Integer
GetTableNameFromId Method	iTableID	Short Integer	String
GetTableNameFromIndex Method	iIndex	Short Integer	String
GetUniqueId Method	None		String
Lock Method	None		Boolean
Open Method	szDatabaseName	String	Void
OpenEx Method	szDatabaseName	String	Void
Unlock Method	None		Boolean
ValidateUser Method	szUser [, szPassword]	String, String	Void

BeginBatchInsert Method

Requires ACT! 4.0.2 or later

Description Puts the current database in batch mode for insertion of new records. Call Lock to lock the current database before using this method. ReturnsTrue if batch mode was successfully started or False if batch mode was not started. Use the LastError property to get information on an error.

Object Database object

Syntax *object*.**BeginBatchInsert** *iBatchFlag*

Parameters *iBatchFlag* A short integer that specifies the batch update mode. Specify 0 for batch updates to a locked database or 1 for batch updates to an unlocked database. Better performance is provided by performing batch updates to a locked database.

Caution: To perform batch inserts to a locked database, use Lock to lock the database and IsLocked to verify that the database was locked before using this method.

Return type Boolean

See also BeginBatchUpdate Method, EndBatchInsert Method, EndBatchUpdate Method, IsInBatchMode Property, IsLocked Property, Lock Method, Unlock Method

LastError Property in common properties and methods

Example This example demonstrates batch mode record insertion on an ACT! database.

```
Dim objContact As Object
Dim dbAct as Object
Dim LoopCount as Integer

Set dbAct.CreateObject("actole.database")
dbAct.Open dbName

Set objContact = dbAct.CONTACT
```

```
If dbAct.lock = False Then
    MsgBox "Could not lock database. Aborting"
    Exit Sub
End If

If dbAct.IsLocked = False Then
    MsgBox "Could not lock database. Aborting"
    Exit Sub
End If

If dbAct.BeginBatchInsert = False then
    MsgBox "Could not turn on batch mode. Error was: " & dbAct.LastError
    Exit sub
End If

If dbAct.IsInBatchMode = False then
    MsgBox "Could not turn on batch mode. Error was: " & dbAct.LastError
    Exit sub
End If

'Update 100 records here.
for LoopCount = 1 to 100
    objContact.Add
    objContact.Data 26, "Contact "
    objContact.Data 25, "Contact Company "
    objContact.Update
Next LoopCount

If dbAct.Unlock = False Then
    MsgBox "Could not unlock database. Aborting"
    Exit Sub
End If

if dbAct.EndBatchInsert = FALSE then
    MsgBox "Could not unlock database. Error was: "& dbAct.LastError
    Exit Sub
End If

dbAct.Close
```

BeginBatchUpdate Method

Requires ACT! 4.0.2 or later

Description Puts the current database in batch mode for updating existing records. ReturnsTrue if batch mode was successfully started or False if batch mode was not started. Use the LastError property to get information on an error.

Object Database object

Syntax *object*.**BeginBatchUpdate** *iBatchFlag*

Parameters *iBatchFlag* A short integer that specifies the batch update mode. Specify 0 for batch updates to a locked database or 1 for batch updates to an unlocked database. Better performance is provided by performing batch updates to a locked database.

Caution: To perform batch updates to a locked database, use Lock to lock the database and IsLocked to verify that the database was locked before using this method.

Return type Boolean

See also BeginBatchInsert Method, BeginBatchUpdate Method, EndBatchInsert Method, EndBatchUpdate Method, IsInBatchMode Property, IsLocked Property, Lock Method, Unlock Method

LastError Property in common database properties and methods

Example

```
' This code demonstrates batch mode record updates on an ACT! database.
Dim objContact As Object
Dim dbAct as Object
Dim LoopCount as Integer
Set dbAct.CreateObject("actole.database")
dbAct.Open dbName

Set objContact = dbAct.CONTACT

If dbAct.Lock = False Then
    MsgBox "Could not lock database. Aborting"
    Exit Sub
End If

If dbAct.IsLocked = False Then
    MsgBox "Could not lock database. Aborting"
    Exit Sub
End If

If dbAct.BeginBatchUpdate 0 = False then
    MsgBox "Could not turn on batch mode. Error was: " & dbAct.LastError
    Exit Sub
End If

If dbAct.IsInBatch = False then
    MsgBox "Could not turn on batch mode. Error was: " & dbAct.LastError
    Exit Sub
End If

'Update 100 records here.
for LoopCount = 1 to 100
    objContact.Edit
    objContact.Data 26, "Contact "
    objContact.Data 25, "Contact Company "
    objContact.Update
Next LoopCount

If dbAct.Unlock = False Then
    MsgBox "Could not unlock database. Aborting"
    Exit Sub
End If

If dbAct.EndBatchUpdate = FALSE then
    MsgBox "Could not unlock database. Error was: "& dbAct.LastError
    Exit Sub
End If
dbAct.Close
```

Close Method

Description Closes the current database. Call this method to close an open database or after unsuccessfully validating a user to a multiuser database. To improve performance by ensuring that all memory is released, always use this method before setting the object to null.

Object Database object

Syntax *object*.**Close**

Return type Void

See also IsOpen Property, IsOpening Property

EndBatchInsert Method

Requires ACT! 4.0.2 or later

Description Turns off batch mode to end batch insertion of new records for the current database. Returns True if batch mode was successfully turned off or False if batch mode was not turned off. Use the LastError property to get information on an error.

Object Database object

Syntax *object*.**EndBatchInsert**

Return type Boolean

See also BeginBatchInsert Method, BeginBatchUpdate Method, EndBatchUpdate Method, IsInBatchMode Property, IsLocked Property, Lock Method, Unlock Method

Example See BeginBatchInsert and BeginBatchUpdate methods.

LastError Property in common properties and methods

EndBatchUpdate Method

Requires ACT! 4.0.2 or later

Description Turns off batch mode to end batch update of existing records for the current database. Returns True if batch mode was successfully turned off or False if batch mode was not turned off. Use the LastError property to get information on an error.

Object Database object

Syntax *object*.**EndBatchUpdate**

Return type Boolean

See also BeginBatchInsert Method, BeginBatchUpdate Method, EndBatchUpdate Method, IsInBatchMode Property, IsLocked Property, Lock Method, Unlock Method

Example See BeginBatchInsert and BeginBatchUpdate methods.

LastError Property in common properties and methods

GetDatabasePath Method

Requires ACT! 4.0 or later

Description Returns a string that contains the full path to the currently open database. If no database is open, a null value or empty string is returned.

Object Database object

Syntax *object*.**GetDatabasePath**

Return type String

Example This example gets the path of the currently open database from the Database object.

```
dim objDatabase as Object

Set objDatabase = CreateObject("ACTOLE.DATABASE")
objDatabase.Open("C:\My Documents\ACT\Database\ACT5demo.dbf")
if objDatabase.IsOpen = FALSE then
   MsgBox "Failed opening the database"
else
   MsgBox "Currently open database is: " & objDatabase.GetDatabasePath
endif

objDatabase.Close
Set objDatabase = Nothing
```

GetTableId Method

Requires ACT! 3.0.6 or later

Description Returns a short integer that contains the Table ID of the ACT! table with the specified index, as stored in the ACT! codebase schema.

Object Database object

Syntax *object*.**GetTableId** *iIndex*

Parameters *iIndex* A short integer that specifies the table index number, in a range between 0 and one less than the value of TableCount.

Return type Short Integer

Comments This method returns the following Table ID values:

Table ID	Table name	Table ID	Table name
1	Contact table	16	Group table
2	Activity table	32	Sales table
4	Notes/History table	64	List table
8	E-mail table		

See also GetTableNameFromId Method, GetTableNameFromIndex Method, TableCount Property

GetTableNameFromId Method

Requires ACT! 3.0.6 or later

Description Returns a string that contains the table name for the specified Table ID.

Object Database object

Syntax *object*.**GetTableNameFromId** *iTableID*

Parameters *iTableID* A short integer that specifies the Table ID.

This method returns the following table names:

Table ID	Table name	Table ID	Table name
1	Contact table	16	Group table
2	Activity table	32	Sales table
4	Notes/History table	64	List table
8	E-mail table		

Return type String

See also GetTableId Method, GetTableNameFromIndex Method, TableCount Property

GetTableNameFromIndex Method

Requires ACT! 3.0.6 or later

Description Returns a string that contains the table name for the specified table index number.

Object Database object

Syntax *object*.**GetTableNameFromIndex** *iIndex*

Parameters *iIndex* A short integer that specifies the table index number, in a range between 0 and one less than the value of TableCount.

Return type String

See also GetTableId Method, GetTableNameFromId Method, TableCount Property

GetUniqueId Method

Description Returns a string that contains a new Unique ID.

Object Database object

Syntax *object*.**GetUniqueId**

Return type String

See also IsOpen Property

Lock Method

Requires ACT! 4.0.2 or later

Description Locks the current database. If the database is currently in use, either on a network drive or the local computer, the database lock is delayed by five minutes. True is returned if the database is successfully locked or False is returned if the database could not be locked. Use the LastError property to get information on an error.

Object Database object

Syntax *object*.**Lock**

Return type Boolean

Comments To use this method, the logged on user must have an Administrator security level.

See also BeginBatchInsert Method, BeginBatchUpdate Method, EndBatchInsert Method, EndBatchUpdate Method, IsInBatchMode Property, IsLocked Property, Unlock Method

LastError Property in common properties and methods

Example See BeginBatchInsert and BeginBatchUpdate methods.

Open Method

Description Opens the specified database. This method opens a single user or multiuser database, and must be successful to access database-dependent properties and methods.

Object Database object

Syntax *object*.**Open** *szDatabaseName*

Parameters *szDatabaseName*A string that specifies the name and path of an ACT! database.

Return type Void

See also IsMultiUser Property, IsOpen Property, IsOpening Property, OpenEx Method, ValidateUser Method

OpenEx Method

Requires ACT! 4.0 or later

Description Opens the specified database, bypassing the validation process provided by the ValidateUser method. To use this method, the specified database must be open in the ACT! application. This method opens a single user or multiuser database, and must be successful to access properties and methods that are database-dependent.

Object Database object

Syntax *object*.**OpenEx** *szDatabaseName*

Parameters *szDatabaseName*A string that specifies the name and path of an ACT! database. Specify szDatabaseName as "" to open the currently open database.

Return type Void

See also IsMultiUser Property, IsOpen Property, IsOpening Property, ValidateUser Method

Example This example opens a database without validating the user.

```
dim objDatabase as Object
Set objDatabase = CreateObject("ACTOLE.DATABASE")
objDatabase.OpenEx ""
if objDatabase.IsOpen = FALSE then
    MsgBox "Failed opening the database"
else
    MsgBox "Currently open database is: " & objDatabase.GetDatabasePath
endif

dim objDatabase as Object
Set objDatabase = Nothing
```

Unlock Method

Requires ACT! 4.0.2 or later

Description Unlocks the current database. True is returned if the database is successfully unlocked or False is returned if the database could not be unlocked. Use the LastError property to get information on an error.

Object Database object

Syntax *object*.**Unlock**

Return type Boolean

Comments To use this method, the logged on user must have an Administrator security level.

See also BeginBatchInsert Method, BeginBatchUpdate Method, EndBatchInsert Method, EndBatchUpdate Method, IsInBatchMode Property, IsLocked Property, Lock Method

LastError Property in common properties and methods

Example See BeginBatchInsert and BeginBatchUpdate methods.

ValidateUser Method

Description Authenticates a user to a multiuser database.

Object Database object

Syntax *object*.**ValidateUser** *szUser* [, *szPassword*]

Parameters *szUser* A string that specifies the user's name.

[*szPassword*] A string that specifies the user's password. This parameter is required if the user has a password.

Return type Void

Comments Call this method after the following conditions have been met:

1 After calling Open, and

2. False is returned for IsOpen, and

3 True is returned for IsOpening or IsMultiUser.

Note: To use this method, the specified user must have previously logged on to the database and generated a My Record contact record. When more than one user is enabled for logon to the database, call this method after calling Open.

See also IsMultiUser Property, IsOpen Property, IsOpening Property, Open Method

Example This sample demonstrates how to properly log on a multiuser ACT! database

```
Dim objDatabase as Object

'Create the object
Set objDatabase = CreateObject("ACTOLE.DATABASE")

'Open the database
objDatabase.Open dbName

'Perform login validation - nothing really happens
If objDatabase.IsMultiUser Then
    'at this point you can display a dialog which asks for
    'username and password. The dialog is something you create.
    objDatabase.ValidateUser username, password

    If objDatabase.Error Then
        Login = False
    Else
        Login = True
    End If
Else
    Login = True
    'Single-user database - validation not required
End If

If objDatabase.IsOpen Then
    'At this point it is now safe to start reading/writing ACT! data.
    objDatabase.Close
End If

Set objDatabase = Nothing
```

Email object

This object contains e-mail information for the active Database object. The Email object contains e-mail information for the active Database object. The following methods apply only to the Email object. See Common properties and methods for additional properties and methods that apply to the Email object.

Methods

Name	Parameter(s)	Parameter type(s)	Return type
ClearContactScope Method	None		Void
SetContactScope Method	szUniqueID	String	Void

ClearContactScope Method

Requires ACT! 3.0.7 or later

Description Clears any existing contact scoping. Resets the current record position to the first record and rebuilds the list of E-mail records without applying contact scoping.

Object Email object

Syntax *object*.**ClearContactScope**

Return type Void

See also SetContactScope Method

SetContactScope Method

Requires ACT! 3.0.7 or later

Description Narrows the current set of E-mail records to the E-mail records for the contact record with the specified Unique ID. You can apply this scoping method along with other scoping methods and filters.

Object Email object

Syntax *object*.**SetContactScope** *szUniqueID*

Parameters *szUniqueID* A string that specifies the Unique ID of a contact record.

Return type Void

See also ClearContactScope Method

Example

```
'List the e-mail address and other e-mail properties for a contact.

Dim objDatabase As Object
Dim objEmail As Object
Dim strContactID As String
'Clear the listbox
lstDisplay.Clear

'Create the object
lstDisplay.AddItem "Creating Database Object"
Set objDatabase = CreateObject("ACTOLE.DATABASE")

'Open the database
lstDisplay.AddItem "Opening Database"
objDatabase.Open dbName
```

```
'Assign the correct table object
Set objEmail = objDatabase.EMAIL

If objDatabase.IsOpen Then
    objEmail.SetContactScope strContactID
    lstDisplay.AddItem objEmail.RecordCount & " found for the Contact"
    Set objEmail = Nothing
    objDatabase.Close
    lstDisplay.AddItem "Database Closed"
End If

Set objDatabase = Nothing
```

ExceptionInfo object

The ExceptionInfo object contains the recurring exception information for a particular activity. The following properties and methods apply only to the ExceptionInfo object. See Common properties and methods for additional properties and methods that apply to this object.

Properties

Name	Parameter(s)	Parameter type(s)	Value type	Access
Count Property	None	Short Integer	Short Integer	Read Only

Count Property

Description Returns the number of exceptions for the current activity.

Object ExceptionInfo object

Syntax *object*.**Count**

Value type Short Integer, Read Only

Comments An exception item is a Date value stored with the primary recurring activity. The Date value represents a recurring instance of the activity that has been altered.

For example, an activity is set to recur every other day, for example the 1st, 3rd, and the 5th, and the user moves the recurring instance for the 3rd to the 4th. ACT! generates a new activity record with a date of the 4th and adds the 4th to the exception list of the primary recurring activity, which is the 1st in this example, thus increasing the count or number of exceptions.

Methods

Name	Parameter(s)	Parameter type(s)	Return type
Add Method	exceptionDate	Date	Void
Clear Method	None		Void
Remove Method	exceptionDate	Date	Void
Seek Method	exceptionDate	Date	Void
Value Method	iIndex	Short Integer	Date/Variant

Add Method

Description Adds a date to the exception list for the current activity.

Object ExceptionInfo object

Syntax *object*.**Add** *exceptionDate*

Parameters *exceptionDate* A Date value that specifies the date to add to the exception list, formatted in Windows Regional Settings Short Date style.

Return type Void

Clear Method

Description Clears and commits the exception list for the current activity. Call this method to erase the list of exceptions for a primary recurring activity.

Object ExceptionInfo object

Syntax *object*.**Clear**

Return type Void

Comments This method does not erase any activity records; it just makes ACT! erase the exception and reset the previous date. For example, if an occurrence of a recurring activity is moved from the 3rd to the 4th, a new activity record is generated for the 4th. If the exception list for the primary recurring activity is erased, the activity for the 3rd reappears.

A recurring activity has only one record, which is the primary activity. All other instances of a recurring activity are mathematically calculated and are not represented in the physical database.

Remove Method

Description Removes a date from the exception list. Removing a date causes the recurring instance for the deleted date to reappear in the calendar as an activity.

Object ExceptionInfo object

Syntax *object*.**Remove** *exceptionDate*

Parameters exceptionDate A Date value that specifies the date to remove from the exception list, formatted in Windows Regional Settings Short Date style.

Return type Void

Seek Method

Description Locates the specified exception date. Call this method to determine if a specified date exists in the exception list.

Object ExceptionInfo object

Syntax *object*.**Seek** *exceptionDate*

Parameters *exceptionDate* A Date value that specifies the exception date to locate, formatted in Windows Regional Settings Short Date style.

Return type Void

Value Method

Description Returns a date value for a position in the exception list.

Object ExceptionInfo object

Syntax *object*.**Value** *iIndex*

Parameters *iIndex* A short integer that specifies the position within the exception list, in the range between 1 and Count.

Return type Date/Variant

See also Count Property

Fields object

The Fields object contains the data properties for a specified ACT! field. The following properties and methods apply only to the Fields object. See Common properties and methods for additional properties and methods that apply to this object.

Properties

Name	Parameter(s)	Parameter type(s)	Value type	Access
AutoPopulate Property	iFieldID [,=TrueIFalse]	Short Integer, Boolean	Boolean	Read/Write
Count Property	None		Short Integer	Read Only
DecimalPlaces Property	iFieldID, [= iDecPlace]	Short Integer Short Integer	Short Integer	Read/Write
EntryRule Property	iFieldID	Short Integer	Short Integer	Read/Write
EntryTrigger Property	iFieldID [, szExecutableName]	Short Integer, String	String	Read/Write
Exists Property	szField	String	Boolean	Read Only
ExitTrigger Property	iFieldID [, szExecutableName]	Short Integer, String	String	Read/Write
FieldId Property	szField	String	Short Integer	Read Only
FieldIdAt Property	iFieldID	Short Integer	Short Integer	Read Only
Flags Property	iFieldID [= lFlag]	Short Integer, Long Integer	Long Integer	Read/Write
HasPopupList Property	iFieldID	Short Integer	Boolean	Read Only
Id Property	iFieldID	Short Integer	Short Integer	Read Only
InitialValue Property	iFieldID	Short Integer	String	Read/Write
IsBlockSync Property	iFieldID [=TrueIFalse]	Short Integer, Boolean	Boolean	Read/Write
IsCutHistory Property	iFieldID [TrueIFalse]	Short Integer, Boolean	Boolean	Read/Write
IsIndexed Property	iFieldID	Short Integer	Boolean	Read Only
IsPrimary Property	iFieldID	Short Integer	Boolean	Read/Write
IsSortable Property	iFieldID	Short Integer	Boolean	Read Only
Label Property	iFieldID [= szLabel]	Short Integer, String	String	Read/Write
Length Property	iFieldID	Short Integer	Short Integer	Read Only
Modifiable Property	iFieldID	Short Integer	Long Integer	Read Only
PopupInfo Property	iFieldID, iPopupIndex	Short Integer, Short Integer	Object/LPDISPATCH	Read Only
Type Property	iFieldID	Short Integer	Short Integer	Read Only

AutoPopulate Property

Requires ACT! 5.0 or later

Description Gets or sets the Automatically Add New Items To Drop-Down option in Define Fields for the specified field. Returns True if the option is selected or False if the option is not selected. If set, AutoPopulate enables auto-population of a field with a drop-down list.

Object Fields object

Syntax *object*.**AutoPopulate** *iFieldID* [, = *True*|*False*]

Parameters *iFieldID* A short integer that specifies the field ID for the field. See ACT! Databases for lists of field IDs and names (Field constants).

[= *True*|*False*] Specify True to select the Automatically Add New ItemsTo Drop-Down option for the specified field or False to deselect it. Omit this optional parameter to determine the setting of the option.

Value type Boolean, Read/Write

Example

```
Set objDatabase = CreateObject("ACTOLE.DATABASE")

'Open the database
List1.AddItem "Opening Database"
objDatabase.Open dbName

If objDatabase.IsOpen Then
    Set objContact = objDatabase.CONTACT
    'Let the User1 fields auto-populate property be set to true
    objContact.fields.AutoPopulate CF_User1, True
    objDatabase.close
End If

Set objDatabase = Nothing
```

Count Property

Description Returns the number of fields in the parent table object.

Object Fields object

Syntax *object*.**Count**

Value type Short Integer, Read Only

See also DecimalPlaces Property, EntryTrigger Property, Exists Property, ExitTrigger Property

DecimalPlaces Property

Description Gets and sets the number of decimal places in the specified field.

Object Fields object

Syntax *object*.**DecimalPlaces** *iFieldID [= iDecPlace]*

Parameters *iFieldID* A short integer that specifies the field ID for the field. See ACT! Databases for lists of field IDs and names (Field constants).

[= *iDecPlace*] A short integer that specifies the number of decimal places to set for the specified field. Omit this optional parameter to get the number of decimal places in the specified field.

Value type Short Integer, Read/Write

See also Count, Modifiable

EntryRule Property

Requires ACT! 4.0.2 or later

Description Gets and sets the entry rule attribute for the field specified by the iFieldID parameter.

Object Fields object

Syntax *object*.**EntryRule** *iFieldID [=iEntryRule]*

Parameters *iFieldID* A short integer that specifies the field ID for the field. See ACT! Databases for lists of field IDs and names (Field constants).

[*=iEntryRule*] A short integer that specifies the entry rule for the specified field. Omit this optional parameter to get the entry rule.

Value type Short Integer, Read/Write

Comments This property can get or set one of the following values:

Value	Description
0	No rules
1	Field cannot be blank (data is required)
2	Only from drop-down (information must be selected from a drop-down list)
3	Protected (protects the field from modification)

See also EntryTrigger Property, IsCutHistory Property

EntryTrigger Property

Description Gets and sets the Entry Trigger executable file for the specified field.

Object Fields object

Syntax *object*.**EntryTrigger** *iFieldID* [*, szExecutableName*]

Parameters *iFieldID* A short integer that specifies the field ID for the field. See ACT! Databases for lists of field IDs and names (Field constants).

[*, szExecutableName*]A string that specifies the executable application (.EXE) or macro (.MPR) entry trigger file to launch when the specified field is entered. Omit this optional parameter to get the name of the executable file that is the entry trigger for the specified field.

Value type String, Read/Write

See also Count Property, ExitTrigger Property

Example

```
dim objDatabase as object
Set objDatabase = CreateObject("ACTOLE.DATABASE")
objDatabase.Open dbName

Dim tableObject As Object
Set tableObject = objDatabase.CONTACT

'*** Set an entry trigger to NOTEPAD.EXE
tableObject.FIELDS.EntryTrigger CF_Company, "notepad.EXE"

'*** Get the entry trigger
dim strOut as String
strOut = tableObject.FIELDS.EntryTrigger (CF_Company)
objDatabase.Close

set objDatabase = Nothing
```

Exists Property

Description Returns True if the specified field exists or False if it does not exist.

Object Fields object

Syntax *object*.**Exists** *szField*

Parameters *szField* A string that specifies the label of the field.

Value type Boolean, Read Only

See also Count Property, Label Property

ExitTrigger Property

Description Gets and sets the Exit Trigger executable file for the specified field.

Object Fields object

Syntax *object*.**ExitTrigger** *iFieldID* [*, szExecutableName*]

Parameters *iFieldID* A short integer that specifies the field ID for the field. See ACT! Databases for lists of field IDs and names (Field constants).

[*, szExecutableName*]A string that specifies the executable application (.EXE) or macro (.MPR) exit trigger file to launch when the specified field is exited. Omit this optional parameter to get the name of the executable file that is the exit trigger for the specified field.

Value type String, Read/Write

See also Count Property, EntryTrigger Property

Example This example demonstrates how to set and get exit triggers.

```
dim objDatabase as object
Set objDatabase = CreateObject("ACTOLE.DATABASE")
objDatabase.Open dbName

Dim tableObject As Object
Set tableObject = objDatabase.CONTACT

'*** Set an exit trigger to NOTEPAD.EXE
tableObject.FIELDS.ExitTrigger CF_Company, "notepad.EXE"

'*** Get the exit trigger
dim strOut as String
strOut = tableObject.FIELDS.ExitTrigger (CF_Company)
objDatabase.Close

set objDatabase = Nothing
```

FieldId Property

Requires ACT! 3.0.7 or later

Description Returns a short integer containing the field ID for the field with the specified name. Field names are shown in Define Fields within the ACT! application.

Object Fields object

Syntax *object*.**FieldId** *szField*

Parameters *szField* A string that specifies the field name. See ACT! Databases for the default field names for fields in ACT! database tables (Field constants).

Value type Short Integer, Read Only

Example Get the field ID for the Company field in the Contact table

```
Dim nFieldId as integer
'nFieldId should have the value 25
nFieldId = ObjContact.FIELDS.FieldId ("Company")
```

FieldIdAt Property

Requires ACT! 3.0.6 or later

Description Returns a short integer containing the field ID for the field with the specified index.

Object Fields object

Syntax *object*.**FieldIdAt** *iFieldID*

Parameters *iFieldID* A short integer that specifies the index number for the field, in the range between 1 and Count.

Value type Short Integer, Read Only

Flags Property

Description Gets and sets the drop-down list flag for the specified field. This property is used to get and set the field definitions for fields with drop-down lists.

Object Fields object

Syntax *object*.**Flags** *iFieldID* [= *lFlag*]

Parameters *iField* A short integer that specifies the field ID for the field. See ACT! Databases for lists of field IDs and names (Field constants).

[= *lFlag*] A long integer that specifies a value between 0 and 3, to set or clear the flag(s) for a specified field. Omit this optional parameter to get the value of the flag.

Value type Long Integer, Read/Write

Comments This property can get or set one of the following values:

Value	Description
0	Clears the flags for Allow Editing and Show Descriptions to disable editing of drop-down list items and disable the display of drop-down list item descriptions for the specified field.
1	Sets the flag for Allow Editing to allow the editing of drop-down list items for the specified field.
2	Sets the flag for Show Descriptions to display drop-down list item descriptions for the specified field.
3	Sets the flags for Allow Editing and Show Descriptions of drop-down list items for the specified field.
5	Sets the flags for Allow Editing and Automatically add new items to drop-down list items for the specified field.
7	Sets the flags for Allow Editing, Automatically add new items, and Show Descriptions to drop-down list items for the specified field.
9	Sets the flags for Allow Editing, Automatically add new items, Show Descriptions and Use drop-down list from: (may be a link from other field).
11	Sets the flags for Allow Editing and Use drop-down list from: (may be a link from other field).
13	Sets the flags for Allow Editing, Automatically add new items, and Use drop-down list from: (may be a link from other field).
15	Sets the flags for Allow Editing, Automatically add new items, Show Descriptions and Use drop-down list from:(may be a link from another field).

HasPopupList Property

Description Returns True if the specified field has a drop-down list or False if the field does not have a drop-down list.

Object Fields object

Syntax *object*.**HasPopupList** *iFieldID*

Parameters *iFieldID* A short integer that specifies the field ID for the field. See ACT! Databases for lists of field IDs and names (Field constants).

Value type Boolean, Read Only

Id Property

Description Gets the field ID of the specified field. Use this property to verify that the specified field exists. If the field does not exist, this property returns the value -1. See ACT! Databases for lists of field IDs and names (Field constants).

Object Fields object

Syntax *object*.**Id** *iFieldID*

Parameters *iFieldID* A short integer that specifies the index number for the field, in the range between 1 and Count.

Value type Short Integer, Read Only

See also Count Property

InitialValue Property

Description Gets and sets the default value in new records for the specified field.

Object Fields object

Syntax *object*.**InitialValue** *iFieldID*

Parameters *iFieldID* A short integer that specifies the field ID for the field. See ACT! Databases for lists of field IDs and names (Field constants).

Value type String, Read/Write

See also Count Property, Modifiable Property

IsBlockSync Property

Description Gets and sets synchronization blocking for the specified field. If True is set or returned, the field is not available (blocked) for data synchronization. If False is set or returned, the field is available for data synchronization.

Object Fields object

Syntax *object*.**IsBlockSync** *iFieldID [= True|False]*

Parameters *iFieldID* A short integer that specifies the field ID for the field. See ACT! Databases for lists of field IDs and names (Field constants).

[=*True*|*False*] Specify True if the field is not available (blocked) for data synchronization or False if the field is available for data synchronization. Omit this parameter to get the synchronization setting for the specified field.

Value type Boolean, Read/Write

See also Count Property, Modifiable Property

IsCutHistory Property

Description Gets and sets the generate history attribute for the specified field. Returns True if a history record is generated when the specified field is modified or False if a history record is not generated. Requires ACT! 4.0.2 or later for the set function.)

Object Fields object

Syntax *object*.**IsCutHistory** *iFieldID* [*True*|*False*]

Parameters *iFieldID* A short integer that specifies the field ID for the field. See ACT! Databases for lists of field IDs and names (Field constants).

True|*False* Specify True to generate a history record when the specified field is modified or False to not generate a history record Requires ACT! 4.0.2 or later. Omit this optional parameter to get the generate history attribute for the specified field.

Value type Boolean, Read/Write

See also Count Property, Modifiable Property

IsIndexed Property

Description Returns True if the specified field has an index applied or False if the field is not indexed.

Object Fields object

Syntax *object*.**IsIndexed** *iFieldID*

Parameters *iFieldID* A short integer that specifies the field ID for the field. See ACT! Databases for lists of field IDs and names (Field constants).

Value type Boolean, Read Only

See also Count Property

IsPrimary Property

Description Returns True if the specified field is a Primary Field or False if the field is not a Primary Field. The contents of primary fields are copied when a contact or group is duplicated.

Object Fields object

Syntax *object*.**IsPrimary** *iFieldID*

Parameters *iFieldID* A short integer that specifies the field ID for the field. See ACT! Databases for lists of field IDs and names (Field constants).

Value type Boolean, Read/Write

See also Count Property, Modifiable Property

IsSortable Property

Description Returns True if the specified field can be indexed or False if it cannot be indexed. Check this property for a field before passing the field to the Sort method.

Object Fields object

Syntax *object*.**IsSortable** *iFieldID*

Parameters *iFieldID* A short integer that specifies the field ID for the field. See ACT! Databases for lists of field IDs and names (Field constants).

Value type Boolean, Read Only

See also Count Property

Label Property

Description Gets and sets the field label that is displayed in the ACT! application. This property refers to the field label that is displayed in the ACT! application and not the database structure field name. Use the Rebuild method to rebuild the table after making changes to field label names.

Caution: This property is modifiable by all users, so be careful when using this as a reference.

Object Fields object

Syntax *object*.**Label** *iFieldID* [= *szLabel*]

Parameters *iFieldID* A short integer that specifies the field ID for the field. See ACT! Databases for lists of field IDs and names (Field constants).

[= *szLabel*] A string that specifies the new label name for the specified field. Omit this optional parameter to get the existing field label for the specified field.

Value type String, Read/Write

See also Count Property, Modifiable Property

Rebuild Method in common properties and methods

Length Property

Description Returns the length of the specified field.

Object Fields object

Syntax *object*.**Length** *iFieldID*

Parameters *iFieldID* A short integer that specifies the field ID for the field. See ACT! Databases for lists of field IDs and names (Field constants).

Value type Short Integer, Read Only

Comments ACT! fields with a length that can be modified return a value of 1 or higher for the length. The ACT! fixed-length system fields of Blob, Date, DateTime, Phone, Time, TimeStamp, and UniqueID return a value of -1 for the length.

See also Count Property, Modifiable Property

Modifiable Property

Description Returns a long integer for the flag that specifies the allowable attribute changes for the field.

Object Fields object

Syntax *object*.**Modifiable** *iFieldID*

Parameters *iFieldID* A short integer that specifies the field ID for the field. See ACT! Databases for lists of field IDs and names (Field constants).

Value type Long Integer, Read Only

See also Count Property

Comments This property can return any combination of the following hexadecimal flag values:

Hexadecimal value	Description
&H0	Field cannot be modified
&H1	Field can be deleted, only by ACT!
&H2	Type can be modified
&H4	Label can be modified
&H8	Length can be modified, only by ACT!

Hexadecimal value	Description
&H10	Decimal places can be modified
&HFFFF	All field attributes are modifiable

To determine if a particular attribute flag is set, AND the hex value of the long integer returned with the desired flag. If the result is equal to the desired flag, the flag is set. If the result is not equal to the desired flag, the flag is not set.

PopupInfo Property

Description Returns a PopupInfo object.

Object Fields object

Syntax *object*.**PopupInfo** *iFieldID*, *iPopupIndex*

Parameters *iFieldID* A short integer that specifies the field ID for the field. See ACT! Databases for lists of field IDs and names (Field constants).

iPopupIndex A short integer that specifies a drop-down list that has been associated with a field. The value of zero is usually returned for this field.

Value type Object/LPDISPATCH, Read Only

Comments This method will initialize a PopupInfo object with the field ID and popup index values, and return its object/dispatch pointer. It is possible to have more than one drop-down list associated with a particular field. Currently, the only field that has more than one drop-down list is the Regarding field in the Activity object.

Example See the PopupCount Property.

Type Property

Description Returns a value that represents the data type for a specified field.

Object Fields object

Syntax *object*.**Type** *iFieldID*

Parameters *iFieldID* A short integer that specifies the field ID for the field. See ACT! Databases for lists of field IDs and names (Field constants).

Value type Short Integer, Read Only

Comments This property returns the following data type values:

Value	Description	Format	Example
0	None		
1	String		fieldName
2	String -Uppercase		FIELDNAME
3	String -Lowercase		fieldname
4	String -Initial Capital		Fieldname
5	Numeric		5
6	Numeric -Currency	($$¢¢)	5000
7	TimeStamp	yyyyMMddhhmmss	19940501202701
8	Date	Short Date style in Windows Regional Settings	11/13/97
9	Time	Time style in Windows Regional Settings	20:27
10	DateTime	Short Date style and Time style in Windows Regional Settings	11/13/97 8:27pm

Value	Description	Format	Example
11	Blob		
12	Binary		
13	Unique ID		7O)E& HP"7R
14	Phone	canonical	+1 (905) 5554993
15	String -URL Address		www.act.com

Note: The field type refers to the data formatting information required by ACT! to display visible fields in the ACT! application. The data may be stored differently in its native format.

Date and time fields must be the required length and be formatted correctly as specified by the Short Date Style and Time Style in Windows Regional Settings when passed as a parameter to the Data property, or an error will be generated. TimeStamp field values are generated by the ACT! application. See Date and time formats for more information.

See also Count Property, Modifiable Property

Data Property, Error Property, LastError Property in common properties and methods

Methods

Name	Parameter(s)	Parameter type(s)	Return type
BeginBatch Method	None		Boolean
EndBatch Method	None		Boolean
GetLinkToList Method	iFieldID iLinkToField iLinkToTable	Short Integer Short Integer Short Integer	Short Integer
SetLinkToList Method	iFieldID, iLinkToField, iLinkToTable	Short Integer, Short Integer, Short Integer	Boolean
UnLinkLists Method	iFieldID, TrueIFalse	Short Integer, Boolean	Boolean

BeginBatch Method

Requires ACT! 4.0.2 or later

Description Prepares the database for schema editing. Returns True if the database is prepared for schema editing and False if it is not prepared. Use the LastError property to get information on an error.

Note: Batch functions must be called in the correct sequence. Any functions called that attempt to place another lock on a schema can cause problems. Therefore, you should call only those functions that are needed during batch mode.

Object Fields object

Syntax *object*.**BeginBatch**

Return type Boolean

See also EndBatch Method

LastError Property in common properties and methods

Example The following sample changes various properties of different fields in the Contact object in Batch mode.

```
Set objDatabase = CreateObject("ACTOLE.DATABASE")
objDatabase .Open dbName
Dim ret As Boolean

If objDatabase .IsOpen Then
    'Create the Contact and Fields objects
    Set objContact = objDatabase .Contact
    Set objCF = objContact.fields

    'Begin the batch processing, catch errors, and exit the process
    'if unsuccessful.
    ret = objCF.BeginBatch

    If ret <> True Then   'If there was an error
        MsgBox "Problem entering batch , Error# " & objCF.LastError
        End Sub
    End If

    'Set some properties for various Contact fields
    objCF.IsCutHistory CF_Name, False
    objCF.IsCutHistory CF_Company, True
    objCF.IsCutHistory CF_City, True
    objCF.SetLinkToList CF_User1, CF_Company, 1
    objCF.EntryRule CF_User2, 2
    objCF.UnLinkLists CF_AltCity, True

    'Complete the batch mode
    ret = objCF.EndBatch
    'If an error occurs, flag the error
    If ret <> True Then
        MsgBox "Problem ending batch , Error# " & objCF.LastError
        End Sub
    End If
End If

Set objCF = Nothing
Set objContact = Nothing
objDatabase.Close
Set objDatabase = Nothing
```

EndBatch Method

Requires ACT! 4.0.2 or later

Description Ends the current batch operation. Returns True if the batch operation has ended and False if it has not ended. Use the LastError property to get information on an error.

Note: Batch functions must be called in the correct sequence. Any functions called that attempt to place another lock on a schema can cause problems. Therefore, you should call only those functions that are needed during batch mode.

Object Fields object

Syntax *object*.**EndBatch**

Return type Boolean

See also BeginBatch Method

LastError Property in common properties and methods

GetLinkToList Method

Requires ACT! 5.0 or later

Description Gets the field ID and table (Contact or Group) with the drop-down list that is linked to by specified field.

Object Fields object

Syntax *object*.**LinkToList** *iFieldID*, *iLinkToField*, *iLinkToTable*

Parameters *iFieldID* A short integer that specifies the field ID of the field for which to find a field with the linked drop-down list the field.

iLinkToField A short integer variable to contain the field ID of the field with the drop-down list linked to the field specified in the iFieldID parameter.

iLinkToTable A short integer variable to contain a value for the table containing the fields with the linked drop-down list. A value of 1 is returned for the Contact table and 16 is returned for the Group table.

See ACT! Databases for lists of field IDs and names (Field constants).

Return type Short Integer

See also SetLinkToList Method, UnLinkLists Method

Example

```
Set objDatabase = CreateObject("ACTOLE.DATABASE")
objDatabase.Open dbName

If objDatabase.IsOpen Then
    Set objContact = objDatabase.CONTACT
    'FieldID: stores the field ID of the field with the drop-down list
    'linked to the CF_User1 field.
    'TableID : Stores the ID of the table (Contact or Group)
    'containing the field with the linked drop-down list.
    objContact.fields.GetLinkToList(CF_User1, FieldID, TableID)
    MsgBox " User1 is using dropdown list from " & FieldID & "  of table " &
        TableID
    Set objContact = Nothing
    objDatabase.close
End If

Set objDatabase = Nothing
```

SetLinkToList Method

Requires ACT! 5.0 or later

Description Sets the field with the drop-down list to link to the specified field in the Contact or Group table. This method locks the current database. If the database is currently in use, either on a network drive or the local computer, the database lock is delayed by five minutes. Returns True if successful or False if unsuccessful.

Object Fields object

Syntax *object*.**SetLinkToList** *iFieldID*, *iLinkToField*, *iLinkToTable*

Parameters *iFieldID* A short integer that specifies the field ID of the field for which to link to the drop-down list in the field specified by the iLinkToField parameter.

iLinkToField A short integer that specifies the field ID of the field with the drop-down list to link to the field specified by the iFieldID parameter.

iLinkToTable A short integer that specifies the table containing the field with the drop-down list to link to the field specified by the iFieldID parameter. Specify 1 for the Contact table or 16 for the Group table.

See ACT! Databases for lists of field IDs and names (Field constants).

Return type Boolean

Example

```
Set objDatabase = CreateObject("ACTOLE.DATABASE")
objDatabase.Open dbName

If objDatabase.IsOpen Then
    Set objContact = objDatabase.CONTACT
    'CF_User1 field to link to the drop-down list in the User1 field
    'of the Group table.
    objContact.fields.SetLinkToList(CF_User1, GF_User1, 16)
    MsgBox " User1 is using drop-down list from " & fieldID & "  of table " &
        TableId
    Set objContact = Nothing
    objDatabase.close
End If

Set objDatabase = Nothing
```

UnLinkLists Method

Requires ACT! 5.0 or later

Description Removes the association of a field drop-down list with another list, so the two lists are independent. This means that a change to the list in a field is not changed in the list in another field. Returns True if successful or False if unsuccessful.

Object Fields object

Syntax *object*.**UnLinkLists** (*iFieldID, True*|*False*)

Parameters *iFieldID* A short integer that specifies the field ID of the field with which to remove the association with the list in another field. See ACT! Databases for lists of field IDs and names (Field constants).

True|*False* Specify True to link the lists and False to unlink the lists.

Return type Boolean

See also GetLinkToList Method, SetLinkToList Method

Example

```
Set objDatabase = CreateObject("ACTOLE.DATABASE")
objDatabase.Open dbName

If objDatabase.IsOpen Then
    Set objContact = objDatabase.CONTACT
    'The User1 fields drop down list should not be linked to another
    'list and copy the current list.
    objContact.fields.UnLinkLists(CF_User1, True)
    Set objContact = Nothing
    objDatabase.close
End If

Set objDatabase = Nothing
```

Group object

The Group object contains group and member information for the active Database object. The following properties and methods apply only to the Group object. See Common properties and methods for additional properties and methods that apply to this object.

Properties

Name	Parameter(s)	Parameter type(s)	Value type	Access
ContactCount Property	None		Long Integer	Read Only
Members Property	None		Object/LPDISPATCH	Read/Write

ContactCount Property

Description Returns the number of contact members for the current group. This property returns the same count as members.

Object Group object

Syntax *object*.**ContactCount**

Value type Long Integer, Read Only

See also RecordCount Property in common properties and methods

Members Property

Description Returns a Members object for the specified group.

Object Group object

Syntax *object*.**Members**

Value type Object/LPDISPATCH, Read/Write

Methods

Name	Parameter(s)	Parameter type(s)	Return type
AddContact Method	szUniqueID	String	Void
AddSubGroup Method	None		Void
AssignParent Method	szUniqueID	String	Boolean
ChangeToParentGroup Method	None		Boolean
ChangeToSubGroup Method	szUniqueID	String	Boolean
ClearContactScope Method	None		Void
GetParent Method	None		String
GetSubGroup Method	iIndex	Short Integer	String
GetSubGroupCount Method	None		Long Integer
GetSubGroupList Method	None		Object/LPDISPATCH
GroupType Method	None		Short Integer
RemoveContact Method	szUniqueID	String	Void
SetContactScope Method	szUniqueID	String	Void

AddContact Method

Description Adds a member to the current group. This method adds the specified contact to the current group if the current record is valid. The group must be in edit or add mode, and the record must be updated before any navigational calls are made.

Object Group object

Syntax *object.***AddContact** *szUniqueID*

Parameters *szUniqueID* A string that specifies the Unique ID of the contact record to add to the current group.

Return type Void

See also ContactCount Property, RemoveContact Method

AddSubGroup Method

Requires ACT! 5.0 or later

Description Creates a subgroup for the current group and leaves the group in add mode. This method works like the Add method, but can be used only for adding subgroups.

Object Group object

Syntax *object.***AddSubGroup**

Return type Void

Comments Creates a new record buffer with default field values so that field data can be assigned using the Data property. The new record is committed using the Update method or aborted if any record navigational method is called.

See also AssignParent Method, ChangeToParentGroup Method, ChangeToSubGroup Method, GetParent Method, GetSubGroup Method, GetSubGroupCount Method, GetSubGroupList Method, GroupType Method

Data Property, Update Method in common properties in methods

Example

```
Set objDatabase = CreateObject("ACTOLE.DATABASE")
'Open the database
objDatabase.Open dbName

If objDatabase.IsOpen Then
    'Create the Group object
    Set objGroup = objDatabase.GROUP
    'Move to the first group in the table
    objGroup.MoveFirst
    'Create a new subgroup for the first group
    guid = objGroup.AddSubGroup

    'Set field data
    objGroup.Data GF_Name, "My Subgroup"

    'Update the contents of the record
    objGroup.Update

    objDatabase.close
End If

'Clear the object
Set objDatabase = Nothing
```

AssignParent Method

Requires ACT! 5.0 or later

Description Sets the parent group of the current subgroup or sets the parent of a parent group that has no subgroups. The group must be in edit or add mode. Returns True if the parent is successfully set or False if unsuccessful.

Object Group object

Syntax *object.***AssignParent** *szUniqueID*

Parameters *szUniqueID* A string that specifies the Unique ID of the parent group record.

Return type Boolean

See also AddSubGroup Method, ChangeToParentGroup Method, ChangeToSubGroup Method, GetParent Method, GetSubGroup Method, GetSubGroupCount Method, GetSubGroupList Method, GroupType Method

Example

```
Set objDatabase = CreateObject("ACTOLE.DATABASE")
'Open the database
objDatabase .Open dbName

If objDatabase .IsOpen Then
    'Create the Group object
    Set objGroup = objDatabase.GROUP
    'Go to the subgroup to which to assign a parent
    objGroup.GoTo subGroupId
    'Edit the record
    objGroup.Edit
    'Assign the parent for that subgroup
    objGroup.AssignParent groupUniqueID
    objGroup.Update
    Set objGroup = Nothing
    objDatabase.close
End If

'Clear the object
Set objDatabase = Nothing
```

ChangeToParentGroup Method

Requires ACT! 5.0 or later

Description Changes the current subgroup to a parent group. The group must be in edit or add mode. Returns True if the subgroup is successfully changed to a parent group or False if unsuccessful.

Object Group object

Syntax *object.***ChangeToParentGroup**

Return type Boolean

See also AssignParent Method, AddSubGroup Method, ChangeToSubGroup Method, GetParent Method, GetSubGroup Method, GetSubGroupCount Method, GetSubGroupList Method, GroupType Method

Example

```
Set objDatabase = CreateObject("ACTOLE.DATABASE")
'Open the database
objDatabase .Open dbName
```

```
If objDatabase .IsOpen Then
    'Create the group object
    Set objGroup = objDatabase.GROUP
    'Go to the subgroup to change to a parent group
    objGroup.GoTo subGroupId
    objGroup.Edit
    'Change the subgroup to a parent group
    objGroup.ChangeToParentGroup
    objGroup.Update
    Set objGroup = Nothing
    objDatabase.close
End If

'Clear the object
Set objDatabase = Nothing
```

ChangeToSubGroup Method

Requires ACT! 5.0 or later

Description Changes the current group to a subgroup of the parent group specified by the szUniqueID parameter. The group must be in edit or add mode. Returns True if the group is successfully changed to a subgroup or False if unsuccessful.

Object Group object

Syntax *object.***ChangeToSubGroup** *szUniqueID*

Parameters szUniqueID A string that specifies the Unique ID of the parent group record.

Return type Boolean

See also AssignParent Method, AddSubGroup Method, ChangeToParentGroup Method, GetParent Method, GetSubGroup Method, GetSubGroupCount Method, GetSubGroupList Method, GroupType Method

Example

```
Set objDatabase = CreateObject("ACTOLE.DATABASE")
'Open the database
objDatabase .Open dbName

If objDatabase .IsOpen Then
    'Create the group object
    Set objGroup = objDatabase.GROUP
    'Go to the subgroup to change to a subgroup
    objGroup.GoTo subGroupId
    'Get the Unique ID parent for that subgroup
    objGroup.Edit
    objGroup.ChangeToSubGroup sParentId
    objGroup.Update
    Set objGroup = Nothing
    objDatabase.close
End If

'Clear the object
Set objDatabase = Nothing
```

ClearContactScope Method

Requires ACT! 3.0.7 or later

Description Clears any existing contact scoping. Resets the current record position to the first record and rebuilds the list of Group records without applying contact scoping.

Object Group object

Syntax *object.***ClearContactScope**

Return type Void

See also SetContactScope Method

GetParent Method

Requires ACT! 5.0 or later

Description Returns a string that specifies the Unique ID of the parent group of the current subgroup.

Object Group object

Syntax *object.***GetParent**

Return type String

See also AssignParent Method, AddSubGroup Method, ChangeToParentGroup Method, ChangeToSubGroup Method, GetSubGroup Method, GetSubGroupCount Method, GetSubGroupList Method, GroupType Method

Example

```
Set objDatabase = CreateObject("ACTOLE.DATABASE")
'Open the database
objDatabase .Open dbName

If objDatabase .IsOpen Then
    'Create the group object
    Set objGroup = objDatabase.GROUP
    'Go to the subgroup to assign to a parent group
    objGroup.GoTo subGroupId
    'Get the Unique ID of the parent for that subgroup
    ParentId = objGroup.GetParent
    'Get the name of the parent group
    objGroup.GoTo ParentId
    List1.AddItem "Parent is " & objGroup.Data(GF_Name)
    Set objGroup = Nothing
    objDatabase.close
End If

'Clear the object
Set objDatabase = Nothing
```

GetSubGroup Method

Requires ACT! 5.0 or later

Description Returns the Unique ID of the specified subgroup of the current group record.

Object Group object

Syntax *object.***GetSubGroup** *iIndex*

Parameters *iIndex* A short integer that specifies the index number of the subgroup. Specify a value between 1 and the value returned by GetSubGroupCount.

Return type String

See also AssignParent Method, AddSubGroup Method, ChangeToParentGroup Method, ChangeToSubGroup Method, GetParent Method, GetSubGroupCount Method, GetSubGroupList Method, GroupType Method

Example

```
Set objDatabase = CreateObject("ACTOLE.DATABASE")
'Open the database
objDatabase.Open dbName

If objDatabase.group.GetSubGroupCount > 0 Then
    'Create the group object
    Set objGroup = objDatabase.GROUP
    'Get the Unique IDs of all subgroups for the group
    For i = 1 To objGroup.GetSubGroupCount
        suid = objGroup.GetSubGroup(i)
        List1.AddItem suid
    Next i
End If
List1.AddItem "Closing Database"
objDatabase.close
Set objDatabase = Nothing
```

GetSubGroupCount Method

Requires ACT! 5.0 or later

Description Returns the number of subgroups for the current parent group.

Object Group object

Syntax *object*.**GetSubGroupCount**

Return type Long Integer

See also AssignParent Method, AddSubGroup Method, ChangeToParentGroup Method, ChangeToSubGroup Method, GetParent Method, GetSubGroup Method, GetSubGroupList Method, GroupType Method

Example

```
Set objDatabase = CreateObject("ACTOLE.DATABASE")
objDatabase.Open dbName

If objDatabase.IsOpen Then
    'Create the group object
    Set objGroup = objDatabase.GROUP
    objGroup.MoveFirst

    'Go through all the groups
    For i = 1 To objGroup.recordCount
        If objGroup.GroupType = 0 Then        'If Parent
            List1.AddItem objGroup.Data(GF_Name) & " is a group"
            'Get the Secondary Group count
            sgrpcount = objGroup.GetSubGroupCount
            List1.AddItem objGroup.Data(GF_Name) & " has " &
                sgroupCount & " subgroups ."
        Else   'SubGroup
            List1.AddItem objGroup.Data(GF_Name) & " is a sub-group"
        End If
        objGroup.MoveNext
    Next i
    objDatabase.close
End If

Set objDatabase = Nothing
```

GetSubGroupList Method

Requires ACT! 5.0 or later

Description Returns a dispatch pointer to an object containing the subgroup records for the current group. Returns NULL on error.

Object Group object

Syntax *object*.**GetSubGroupList**

Return type Object/LPDISPATCH

See also AssignParent Method, AddSubGroup Method, ChangeToParentGroup Method, ChangeToSubGroup Method, GetParent Method, GetSubGroup Method, GetSubGroupCount Method, GroupType Method

Example The following code goes through all the groups. If a group is a parent and the number of subgroups > 0, then create the subgroup list and enumerate the subgroups.

```
Set objDatabase = CreateObject("ACTOLE.DATABASE")
objDatabase.Open dbName
If objDatabase.IsOpen Then
    'Create the Group object
    Set objGroup = base.group
    objGroup.MoveFirst
    For i = 1 To objGroup.recordCount
        'If the group is a parent
        If objGroup.GroupType = 0 Then
            'If the number of subgroups is > 0
            If objGroup.GetSubGroupCount > 0 Then
                'Create another object that contains the subgroups for that
                'particular group.
                Set objSGroup = objGroup.GetSubGroupList
                'Enumerate the subgroups
                objSGroup.MoveFirst
            For j = 1 To objSGroup.recordCount
                List1.AddItem objSGroup.Data(GF_Name)
                objSGroup.MoveNext
            Next j
                Set objSGroup = nothing
            End If
        End If
End If
```

GroupType Method

Requires ACT! 5.0 or later

Description Determines the group type of the current group. Returns 0 if the current group is a parent group or 1 if it is a subgroup.

Object Group object

Syntax *object*.**GroupType**

Return type Short Integer

See also AssignParent Method, AddSubGroup Method, ChangeToParentGroup Method, ChangeToSubGroup Method, GetParent Method, GetSubGroup Method, GetSubGroupCount Method, GetSubGroupList Method

Example

```
Set objDatabase = CreateObject("ACTOLE.DATABASE")
objDatabase.Open dbName
```

```
If objDatabase.IsOpen Then
    'Create the group object
    Set objGroup = objDatabase.GROUP
    objGroup.MoveFirst
    'Go through all the groups
    For i = 1 To objGroup.recordCount

        If objGroup.GroupType = 0 Then        'If a parent
            List1.AddItem objGroup.Data(GF_Name) & " is a group"
            'Get the secondary group count
            sgrpcount = objGroup.GetSubGroupCount
            List1.AddItem objGroup.Data(GF_Name) & " has " &
                sgroupCount & " subgroups ."
        Else        'Subgroup
            List1.AddItem objGroup.Data(GF_Name) & " is a sub-group"
        End If
        objGroup.MoveNext
    Next i
    objDatabase.close
End If

Set objDatabase = Nothing
```

RemoveContact Method

Description Removes the specified contact from being a member of the current group. This method does not delete the contact record from the database; it only removes the relationship between the contact and the group. The group must be in edit or add mode, and the record must be updated before any navigational calls are made.

Object Group object

Syntax *object*.**RemoveContact** *szUniqueID*

Parameters *szUniqueID* A string that specifies the Unique ID of the contact record to remove from the current group.

Return type Void

See also AddContact Method

Error Property, LastError Property in common properties and methods

SetContactScope Method

Requires ACT! 3.0.7 or later

Description Narrows the current set of Group records to the Group records for the contact record with the specified Unique ID. You can apply this scoping method along with other scoping methods and filters.

Object Group object

Syntax *object*.**SetContactScope** *szUniqueID*

Parameters *szUniqueID* A string that specifies the Unique ID of a contact record.

Return type Void

See also ClearContactScope Method

Example List all the groups a contact belongs to.

```
Dim objDatabase As Object
Dim objGroup As Object
Dim strContactID As String
```

```
'Clear the listbox
lstDisplay.Clear

'Create the object
lstDisplay.AddItem "Creating Database object"
Set objDatabase = CreateObject("ACTOLE.DATABASE")

'Open the database
lstDisplay.AddItem "Opening Database"
objDatabase.Open dbName

'Assign the correct table object
Set objGroup = objDatabase.GROUP

If objDatabase.IsOpen Then
    objGroup.SetContactScope strContactID
    lstDisplay.AddItem objGroup.recordCount & " found for the Contact"
    Set objGroup = Nothing
    objDatabase.Close
    lstDisplay.AddItem "Database Closed"
End If

Set objDatabase = Nothing
```

ListTable object

The ListTable object, which requires ACT! 5.0 or later, contains product, product type, and main competitor data for use by the Sales object. The following methods apply only to the ListTable object. See Common properties and methods for additional properties and methods that apply.

Methods

Name	Parameter(s)	Parameter type(s)	Return type
ClearScope Method	None		Void
GetScope Method	None		Short Integer
SetScope Method	iScopeType	Short Integer	Void

ClearScope Method

Requires ACT! 5.0 or later

Description Clears the current scope set on the table, resets the current record position to the first record, and rebuilds the list of ListTable records without any scoping.

Object ListTable object

Syntax *object*.**GetScope**

Return type Void

See also GetScope Method, SetScope Method

Example

```
Set objDatabase = CreateObject("ACTOLE.DATABASE")
objDatabase.Open dbName

If objDatabase.IsOpen Then
    Set objList = objDatabase.ListTable
```

```
    'Get the current scoping on the List table.
    'If it is not 0 (no scoping), then clear the current scope.
    If objList.GetScope <> 0 then
        objList.ClearScope
    End if
Set objList = Nothing
End if
Set objDatabase = Nothing
```

GetScope Method

Requires ACT! 5.0 or later

Description Returns a short integer specifying the current scope type set on the table, such as Product, Product Type, or Competitor.

Object ListTable object

Syntax *object*.**GetScope**

Return type Short Integer

Comments The following values are returned by this method:

Value	Scope type	Value	Scope type
0	No Type	2	Product Type
1	Product	3	Competitor

See also ClearScope Method, SetScope Method

Example

```
Set objDatabase = CreateObject("ACTOLE.DATABASE")

objDatabase.Open dbName
If objDatabase.IsOpen Then
    Set objList = objDatabase.ListTable
    'Get the current scoping on the List table.
    'If it is not 0 (no scoping), then clear the current scope.
    If objList.GetScope <> 0 then
        objList.ClearScope
    End if
Set objList = Nothing
End if

Set objDatabase = Nothing
```

SetScope Method

Requires ACT! 5.0 or later

Description Sets the scope to only read a specific type of sales data, such as Product,Type, or Competitor.

Object ListTable object

Syntax *object*.**SetScope** (*iScopeType*)

Parameters *iScopeType* A short integer specifying the type of sales data to read.

Following is a list of values for this parameter:

Value	Scope type	Value	Scope type
0	No Type	2	Product Type

Value	Scope type	Value	Scope type
1	Product	3	Competitor

Return type Void

See also ClearScope Method, GetScope Method

Example

```
Set objDatabase = CreateObject("ACTOLE.DATABASE")
objDatabase.Open dbName

If objDatabase.IsOpen Then
    Set objList = objDatabase.ListTable
    'Set the scope to product.
    objList.SetScope 1
    List1.AddItem  "Products in ListTable are"
    For lcounter = 1 To objList.recordCount
        List1.AddItem objList.Data (LTF_Name)
        objList.MoveNext
    Next lcounter
End if

Set objList = Nothing
Set objDatabase = Nothing
```

Members object

The Members object is a list of all contacts in a group, automatically generated each time the SDK accesses a group record. Information in the Members object is invalid if the number of contacts in a group is 0, or if a record in the list has been moved since the object was generated. For example, if the SDK calls Group.Members and then Group.MoveNext, the information returned by Group.Members is out of date and must be regenerated. If the SDK calls Group.MoveNext, and then Group.Members, the Members object contains accurate information.

Sample Code

The following sample code demonstrates how to use the Members object:

```
'Create the object
Set objDatabase = CreateObject("ACTOLE.DATABASE")

'Open the database
objDatabase.Open dbName

'Assign the correct table object
Dim tableObject As Object
Set tableObject = objDatabase.CONTACT

If objDatabase.IsOpen Then
    objDatabase.Group.MoveFirst() 'move to the first group record.

    'The contact count
    If objDatabase.Group.ContactCount > 0 Then

        'Define an object variable to access the members object
        dim members As Object
        Set members = objDatabase.Group.Members
```

```
        'Enumerate all contacts
        For Index = 1 To objDatabase.Group.ContactCount
            Dim uniqueID As String
            uniqueID = members.uniqueID

            'Seek to actual members Contact record and display information
            objDatabase.CONTACT.GoTo (uniqueID)

            For lCounter = 1 To (tableObject.FieldCount)
                length = Len(tableObject.Data
                ( tableObject.FIELDS.fieldIDAt(1Counter)))
                If (length > 0) Then
                MsgBox tableObject.Data
                (tableObject.FIELDS.fieldIDAt(lCounter))
                End If
            Next
        Next Index
    End If
End If
```

Note This example assumes that you are logging on to a single-user database.

Properties

The following properties apply only to the Members object. See "Common properties and methods" on page 34 for additional properties and methods that apply to this object.

Name	Parameter(s)	Parameter type(s)	Value type	Access
Name Property	None		String	Read Only
UniqueId Property	None		String	Read Only

Name Property

Description Returns the contact name of the group member.

Object Members object

Syntax *object*.**Name**

Value type String, Read Only

UniqueId Property

Description Returns the contact Unique ID of the group member.

Object Members object

Syntax *object*.**UniqueId**

Value type String, Read Only

NoteHistory object

The NoteHistory object contains notes and historical information for the active Database object. The following methods apply only to the NoteHistory object. See “Common properties and methods” on page 34 for additional properties and methods that apply to this object.

History types

When writing to the type field of the NoteHistory object, use one of the following ID types to generate a history item.

ID type	Field name	Regarding text
0	HT_CallAttempt	Call Attempted
1	HT_CallComplete	Call Completed
2	HT_CallReceived	Call Received
3	HT_FieldChanged	Field Changed
5	HT_LetterSent	Letter Sent
6	HT_MeetingDone	Meeting Held
7	HT_MeetingNotDone	Meeting Not Held
8	HT_ToDoDone	To-do Done
9	HT_ToDoNotDone	To-do Not Done
10	HT_Timer	Timer
11	HT_CallErased	Call Erased
12	HT_DeletedContact	Contact Deleted
13	HT_UpdateContact	Update Contact
14	HT_UpdateConAct	Update Activity
15	HT_UpdateDelAct	Delete Activity
16	HT_MsgMailSent	E-mail Sent
17	HT_CallLeftMessage	Call Left Message
50	HT_FaxSent	Fax Sent
51	HT_SyncSent	Sent Sync
52	HT_SyncReceived	Received Sync
53	HT_ReplaceFieldsLog	Replace Fields Log
54	HT_TODOErased	To-do Erased
55	HT_MeetingErased	Meeting Erased
56	HT_Error	Error
57	HT_SaleWon	Closed/Won Sale
58	HT_SaleLost	Lost Sale
100	HT_Note	Note
101	HT_Attachment	Attachment
102	HT_Email	E-mail

Methods

Name	Parameter(s)	Parameter type(s)	Return type
ClearAttachmentFilter Method	None		Void
ClearContactScope Method	None		Void
ClearGroupScope Method	None		Void
ClearHistoryFilter Method	None		Void
ClearNoteFilter Method	None		Void
SetAttachmentFilter Method	None		Void
SetContactScope Method	szUniqueID	String	Void
SetGroupScope Method	szUniqueID	String	Void
SetHistoryFilter Method	None		Void
SetNoteFilter Method	None		Void

ClearAttachmentFilter Method

Description Clears an existing attachment filter and rebuilds a list of Notes/History records that include any Attachment type records.

Object NoteHistory object

Syntax *object*.**ClearAttachmentFilter**

Return type Void

See also SetAttachmentFilter Method

ClearContactScope Method

Description Clears any existing contact scoping. Resets the current record position to the first record and rebuilds the list of Notes/History records without applying contact scoping.

Object NoteHistory object

Syntax *object*.**ClearContactScope**

Return type Void

See also SetContactScope Method

ClearGroupScope Method

Description Clears any existing group scoping. Resets the current record position to the first record and rebuilds a list of Notes/History records without applying group scoping.

Object NoteHistory object

Syntax *object*.**ClearGroupScope**

Return type Void

See also SetGroupScope Method

ClearHistoryFilter Method

Description Clears an existing history filter. Clears an existing history filter and rebuilds a list of Notes/History records that include History type records.

Object NoteHistory object

Syntax *object*.**ClearHistoryFilter**

Return type Void

See also SetHistoryFilter Method

ClearNoteFilter Method

Description Clears an existing note filter and rebuilds a list of Notes/History records that include Note type records.

Object NoteHistory object

Syntax *object*.**ClearNoteFilter**

Return type Void

See also SetNoteFilter Method

SetAttachmentFilter Method

Description Narrows the current set of Notes/History records by filtering out Attachment type Notes/History records. You can apply this scoping method along with other scoping methods and filters.

Object NoteHistory object

Syntax *object*.**SetAttachmentFilter**

Return type Void

See also ClearAttachmentFilter Method

SetContactScope Method

Description Narrows the current set of Notes/History records to the Notes/History records for the contact record with the specified Unique ID. You can apply this scoping method along with other scoping methods and filters.

Object NoteHistory object

Syntax *object*.**SetContactScope** *szUniqueID*

Parameters *szUniqueID* A string that specifies the Unique ID of a contact record.

Return type Void

See also ClearContactScope Method

Example

```
dim objDatabase as object
Set objDatabase = CreateObject("ACTOLE.DATABASE")
objDatabase.Open dbName

'Get a contact record Unique ID
dim uniqueID as string
objDatabase.CONTACT.MoveFirst
uniqueID = objDatabase.CONTACT.DATA(CF_UniqueID)

'Apply contact scoping to current list of Notes and Histories
objDatabase.NOTEHISTORY.MoveFirst
objDatabase.NOTEHISTORY.SetContactScope(uniqueID)

objDatabase.Close
set objDatabase = Nothing
```

SetGroupScope Method

Description Narrows the current set of Notes/History records to the Notes/History records for the group with the specified Group record Unique ID. You can apply this scoping method along with other scoping methods and filters.

Object NoteHistory object

Syntax *object*.**SetGroupScope** *szUniqueID*

Parameters *szUniqueID* A string that specifies the Unique ID of a group record.

Return type Void

See also ClearGroupScope Method

SetHistoryFilter Method

Description Narrows the current set of Notes/History records by filtering out History type Notes/History records. You can apply this scoping method along with other scoping methods and filters.

Object NoteHistory object

Syntax *object*.**SetHistoryFilter**

Return type Void

See also ClearHistoryFilter Method

SetNoteFilter Method

Description Narrows the current set of Notes/History records by filtering out Note type Notes/History records. You can apply this scoping method along with other scoping methods and filters.

Object NoteHistory object

Syntax *object*.**SetNoteFilter**

Return type Void

See also ClearNoteFilter Method

PopupInfo object

The PopupInfo object contains the popup information for a field. The following properties and methods apply only to the PopupInfo object. See Common properties and methods for additional properties and methods that apply to this object.

Properties

Name	Parameter(s)	Parameter type(s)	Value type(s)	Access
PopupCount Property	None		Short Integer	Read Only

PopupCount Property

Description Returns the number of popup items in a PopupInfo object.

Object PopupInfo object

Syntax *object*.**PopupCount**

Value type Short Integer, Read Only

Example This example demonstrates how to retrieve popup information.

```
dim objDatabase as object
Set objDatabase = CreateObject("ACTOLE.DATABASE")
objDatabase.Open dbName
    'Must include full path and file extension to work
    'For multiuser databases, you must call ValidateUser as well to log on

Dim tableObject As Object
Set tableObject = objDatabase.ACTIVITY

dim PopupInfo as object
set PopupInfo = tableObject.Fields.PopupInfo(AF_REGARDING,0)
    'Must be a valid field value

Dim I as Integer
nCount = PopupInfo.PopupCount
for I = 0 to (nCount - 1)
    list1.additem "Remark: " & PopupInfo.Remark(I)
    list1.additem "Value: " & PopupInfo.Value(I)
next I

set PopupInfo = Nothing
objDatabase.Close
set objDatabase = Nothing
```

Methods

Name	Parameter(s)	Parameter type(s)	Return type
Add Method	szValue [, szRemark]	String, String	Void
Clear Method	None		Void
Remark Method	iIndex	Short Integer	String
Remove Method	iIndex	Short Integer	Void
Value Method	iIndex	Short Integer	String

Add Method

Description Adds a popup item to a field that has the popup type turned on. If a field does not have the Allow Editing option for the drop-down list selected, it is not possible to add a drop-down list item.

Caution: The database should be locked before using this method; otherwise, an error is returned.

Object PopupInfo object

Syntax *object*.**Add** szValue [, *szRemark*]

Parameters *szValue* A string that specifies the value of the popup item.

[*, szRemark*] A string that specifies the popup item's remark, which is the text listed to provide information about the value. For example, the value of CA has the remark text of California. Omit optional parameter to get the remark for the popup item.

Return type Void

Clear Method

Description Clears all the items in a drop-down list, resulting in an empty list.

Caution: The database should be locked before using this method; otherwise, an error is returned.

Object PopupInfo object

Syntax *object*.**Clear**

Return type Void

Remark Method

Description Returns a string of text containing the remark for the drop-down list item with the specified index. The remark provides information about the popup item. For example, the value of CA has the remark text of California.

Object PopupInfo object

Syntax *object*.**Remark** *iIndex*

Parameters *iIndex* A short integer that specifies the index number for the drop-down list item, in the range from 0 to one less than the value of PopupCount.

Return type String

Example See PopupCount Property.

Remove Method

Description Removes the popup item with the specified index from a drop-down list.

Caution: The database should be locked before using this method; otherwise, an error is returned.

Object PopupInfo object

Syntax *object*.**Remove** *iIndex*

Parameters *iIndex* A short integer that specifies the index number for the popup item, in the range from 0 to one less than the value of PopupCount.

Return type Void

Example See PopupCount Property.

Value Method

Description Returns a string containing the drop-down list item value for the popup item with the specified index number. The value in a drop-down list item is the data that is placed into a field in ACT! when an item is chosen from the drop-down list. For example, the value of CA, which is place into a field, has the remark text of California.

Object PopupInfo object

Syntax *object*.**Value** *iIndex*

Parameters *iIndex* A short integer that specifies the index number for the popup item, in the range from 0 to PopupCount.

Return type String

Relations object

The Relations object contains relational information between table objects of the active Database object. The following properties and methods apply only to the Relations object. See Common properties and methods for additional properties and methods that apply to this object.

Sample code

The following sample code demonstrates how to use the Relations object:

```
'The following code lists the different methods in the Relations object.
Dim objDatabase As Object
Dim lcounter As Long

'Clear the listbox
List1.Clear

'Create the object
List1.AddItem "Creating Database object"
Set objDatabase = CreateObject("ACTOLE.DATABASE")

'Open the database
List1.AddItem "Opening Database"
objDatabase.Open dbName
List1.AddItem "Is Open: " & objDatabase.IsOpen

'Assign the correct table object
Dim objRelation As Object
Dim iRelCount As Integer
Dim i As Integer


Set objRelation = objDatabase.Relations
'Get the number of relations in the object
iRelCount = objRelation.Count
List1.AddItem "No Of relations: " & iRelCount

    If objDatabase.IsOpen Then
        For i = 0 To iRelCount - 1
            List1.AddItem objRelation.GetTable1ID(i) & ":" &
                objRelation.GetColumn1ID(i) & ":" &
                objRelation.GetTable2ID(i)   & ":" &
                objRelation.GetColumn2ID(i)  & ":" &
                objRelation.GetRelationType(i)  & ":" &
                objRelation.UsesRelationTable(i)
        Next i
        List1.AddItem "Closing Database"
        Set objRelation = Nothing
        objDatabase.Close
        List1.AddItem "Database Closed"
    End If

'Clear the object
Set objDatabase = Nothing
List1.SetFocus
```

Properties

Name	Parameter(s)	Parameter type(s)	Value type	Access
Count Property	None		Short Integer	Read Only

Count Property

Requires ACT! 3.0.6 or later

Description Returns the number of relations in the Relations object.

Object Relations object

Syntax *object*.**Count**

Value type Short Integer, Read Only

Methods

Name	Parameter(s)	Parameter type(s)	Return type
GetColumn1ID Method	iIndex	Short Integer	Short Integer
GetColumn2ID Method	iIndex	Short Integer	Short Integer
GetRelationTable	iIndex	Short Integer	Short Integer
GetRelationType Method	iIndex	Short Integer	Short Integer
GetTable1ID Method	iIndex	Short Integer	Short Integer
GetTable2ID Method	iIndex	Short Integer	Short Integer
UsesRelationTable Method	iIndex	Short Integer	Boolean

GetColumn1ID Method

Requires ACT! 3.0.6 or later

Description Returns a short integer that contains the value in Column1ID of the first table, for the item with the specified index number.

Object Relations object

Syntax *object*.**GetColumn1ID** *iIndex*

Parameters *iIndex* A short integer that specifies the index number of the item, in the range from 0 to 1 less than the value of Count.

Return type Short Integer

GetColumn2ID Method

Requires ACT! 3.0.6 or later

Description Returns a short integer for the value in Column1ID of the second table, for the item with the specified index number.

Object Relations object

Syntax *object*.**GetColumn2ID** *iIndex*

Parameters *iIndex* A short integer that specifies the index of the item, in the range from 0 to 1 less than the value of Count.

Return type Short Integer

GetRelationType Method

Description Returns a short integer that contains the relation type for the object table, for the item with the specified index number.

Object Relations object

Syntax *object*.**GetRelationType** *iIndex*

Parameters *iIndex* A short integer that specifies the index number of the item, in the range from 0 to 1 less than the value of Count.

Return type Short Integer

Comments The value returned by this property represents the following table relationships:

Value	Relationship	Value	Relationship
1	One_To_One	3	Many_To_One
2	One_To_Many	4	Many_To_Many

GetTable1ID Method

Requires ACT! 3.0.6 or later

Description Returns a short integer that contains the value in Table1ID for the first table.

Object Relations object

Syntax *object*.**GetTable1ID** *iIndex*

Parameters *iIndex* A short integer that specifies the index number of the item, in the range from 0 to 1 less than the value of Count.

Return type Short Integer

GetTable2ID Method

Requires ACT! 3.0.6 or later

Description Returns a short integer for the value in Table1ID for the second table, for the item with the specified index number.

Object Relations object

Syntax *object*.**GetTable2ID** *iIndex*

Parameters *iIndex* A short integer that specifies the index number of the item, in the range from 0 to 1 less than the value of Count.

Return type Short Integer

UsesRelationTable Method

Requires ACT! 3.0.6 or later

Description Returns True if the relation uses the Relational table or False if it does not use the Relational table.

Object Relations object

Syntax *object*.**UsesRelationTable** *iIndex*

Parameters *iIndex* A short integer that specifies the index number of the item, in the range from 0 to 1 less than the value of Count.

Return type Boolean

Comments The Relational table has three fields: Reltype, ColumnID1, ColumnID2. Most of the many-to-many relations are stored as records in the Relational table. Relationships between Group table items and Activity table items, for example, are not stored in the Relation table. There are methods in the table objects to set the scope with respect to another table. These methods use the information in the Relation table to set the scope.

Example See Sample code.

Sales object

The Sales object, which requires ACT! 5.0 or later, contains data for sales process management for the active Database object. The following methods apply only to the Sales object. See Common properties and methods for additional properties and methods that apply to this object.

Methods

Name	Parameter(s)	Parameter type(s)	Return type
AssociateWithContact Method	szUniqueID	String	Void
AssociateWithGroup Method	szUniqueID	String	Void
CompleteSale Method	iStatus closeDate szReason	Short Integer Date String	Void
ReopenSale Method	None		Void

AssociateWithContact Method

Requires ACT! 5.0 or later

Description Associates the current sales record with the specified contact.

Object Sales object

Syntax *object.***AssociateWithContact** (*szUniqueID*)

Parameters *szUniqueID* A string that specifies the Unique ID of a contact record.

Return type Void

See also AssociateWithGroup Method, CompleteSale Method, ReopenSale Method

Example

```
Set objDatabase = CreateObject("ACTOLE.DATABASE")
objDatabase.Open dbName

If objDatabase.IsOpen Then
    Set objSales = objDatabase.SALES
    'New Sales record
    objSales.Add
    objSales.Data SLF_SaleStartDate, startDate
    objSales.Data SLF_SaleDate, closedate
    'These need to be IDs from the List table
    objSales.Data SLF_ProductId, sProductId
    'These need to be IDs from the List table
    objSales.Data SLF_TypeId, sTypeId
    objSales.Data SLF_Units, 100
    objSales.Data SLF_UnitPrice,2.25
    objSales.Data SLF_Amount, 225.00
    objSales.Data SLF_Probability, 65
```

```
    'These need to be IDs from the List table
    objSales.Data SLF_CompetitorId, CompetitorId
    objSales.Data SLF_SalesStage, " Confirmed order on phone,
        waiting for written order"

    'Update the contents of the record
    uniqueID = objSales.Update
    objSales.GoTo (uniqueID)

    If objSales.Error Then
        MsgBox objSales.LastError
    End If

'Make the association between a contact and an activity
objSales.AssociateWithContact sContactId
objSales.AssociateWithGroup sGroupId
Set objSales = Nothing

End If
Set objDatabase = Nothing
```

AssociateWithGroup Method

Requires ACT! 5.0 or later

Description Associates the current sales record with the specified group.

Object Sales object

Syntax *object.***AssociateWithGroup** (*szUniqueID*)

Parameters *szUniqueID* A string that specifies the Unique ID of a group record.

Return type Void

See also AssociateWithContact Method, CompleteSale Method, ReopenSale Method

Example

```
Set objDatabase = CreateObject("ACTOLE.DATABASE")
objDatabase.Open dbName

If objDatabase.IsOpen Then
    Set objSales = objDatabase.SALES
    'New Sales record
    objSales.Add
    objSales.Data SLF_SaleStartDate, startDate
    objSales.Data SLF_SaleDate, closedate
    'These need to be IDs from the List table
    objSales.Data SLF_ProductId, sProductId
    'These need to be IDs from the List table
    objSales.Data SLF_TypeId, sTypeId
    objSales.Data SLF_Units, 100
    objSales.Data SLF_UnitPrice,2.25
    objSales.Data SLF_Amount, 225.00
    objSales.Data SLF_Probability, 65
    'These need to be IDs from the List table
    objSales.Data SLF_CompetitorId, CompetitorId
    objSales.Data SLF_SalesStage, " Confirmed order on phone,
        waiting for written order"

    'Update the contents of the record
    uniqueID = objSales.Update
    objSales.GoTo (uniqueID)
```

```
    If objSales.Error Then
        MsgBox objSales.LastError
    End If

'Make the association between a contact and an activity
objSales.AssociateWithContact sContactId
objSales.AssociateWithGroup sGroupId
Set objSales = Nothing

End If
Set objDatabase = Nothing
```

CompleteSale Method

Requires ACT! 5.0 or later

Description Completes the current sale.

Object Sales object

Syntax *object*.**CompleteSale** (*iStatus, closeDate, szReason*)

Parameters *iStatus* A short integer that specifies the status of the sale.

Following is a list of values for this parameter:

Value	Status	Value	Status
0	Open	2	Lost
1	Won		

closeDate A Date value that specifies the date that the sale closed, formatted in Windows Regional Settings Short Date style.

szReason A string with a maximum length of 65 characters that specifies why the sale was won or lost.

Return type Void

See also AssociateWithGroup Method, AssociateWithGroup Method, ReopenSale Method

Example The following code goes to the required sales record, checks to ensure it is a Sales Opportunity, then completes it as a Won 'sale with today's date as the close date and "Liked the sample and price" as the reason.

```
Dim objDatabase As Object
Dim objSales As Object
Dim suid as string

'Create the Database object and open a database
Set objDatabase = CreateObject("ACTOLE.DATABASE")
objDatabase.Open dbName

If objDatabase.IsOpen Then
    Set objSales = objDatabase.SALES

    objSales.goto suid   'Go to the Sales record you want to complete
    'If the status of the sale is a Sales Opportunity, then complete it
    If objSales.Data(SLF_Status) = "Sales Opportunity" Then
        'Complete it as a Closed/Won sale, with today's date as the
        'close date and the reason is "Liked the sample and price
        objSales.CompleteSale 1, Now, "Liked the Sample and price "
```

```
        'Check for errors
        If objSales.Error Then
            MsgBox objSales.LastError
        End If
    End If

'Close the objects
Set objSales = Nothing
objDatabase.close

End If

'Clear the object
Set objDatabase = Nothing
```

ReopenSale Method

Requires ACT! 5.0 or later

Description Reopens the current sale, setting the status to open.

Object Sales object

Syntax *object*.**ReopenSale**

Return type Void

See also AssociateWithGroup Method, AssociateWithGroup Method, CompleteSale Method

Example The following code goes to the required sales record, checks to ensure it is not a Sales Opportunity, then reopens it.

```
Dim objDatabase As Object
Dim objSales As Object
Dim suid as string

'Create the Database object and open a database
Set objDatabase = CreateObject("ACTOLE.DATABASE")
objDatabase.Open dbName

If objDatabase.IsOpen Then
    Set objSales = objDatabase.SALES
    objSales.goto suid   'Go to the Sales record you want to reopen
    'If the status of the sale is not a Sales Opportunity, then reopen it
    If objSales.Data(SLF_Status) <> "Sales Opportunity" Then
        objSales.ReopenSale
        'Check for errors
        If objSales.Error Then
            MsgBox objSales.LastError
        End If
    End If

'Close the objects
Set objSales = Nothing
objDatabase.close

End If

'Clear the object
Set objDatabase = Nothing
```

Users object

The Users object contains the user information for the active Database object. The following properties and methods apply only to the Users object. See Common properties and methods for additional properties and methods.

Properties

Name	Parameter(s)	Parameter type(s)	Value type	Access
Access Property	iUser [= lType]	Short Integer, Long Integer	Long Integer	Read/Write
Count Property	None		Short Integer	Read Only
Exists Property	szName	String	Boolean	Read Only
Name Property	iUser	Short Integer	String	Read Only
Security Property	iUser [= lValue]	Short Integer, Long Integer	Long Integer	Read/Write
UniqueId Property	iUser	Short Integer	String	Read Only

Access Property

Description Gets and sets the access type of the specified user for the open database.

Object Users object

Syntax *object*.**Access** *iUser* [= *lType*]

Parameters *iUser* A short integer that specifies the ordinal user number, in the range between 1 and Count.

[= *lType*] A long integer that specifies the access type, in the range of 1 to 5. Omit this optional parameter to get the access type for the specified user.

Value type Long Integer, Read/Write

Comments This property returns or sets one of the following values for the access type of the specified user of the open database.

Value	Description
1	Logon access to the database is enabled for the selected user. Synchronization access is not enabled for the selected user.
2	Synchronization access to the database is enabled for the specified user. Logon access is not enabled for the selected user.
3	No access to the database is enabled for the selected user.
4	Direct synchronization access to the database has been set up for the selected user.
5	Logon access and synchronization access to the database is enabled for the selected user.

See also Count Property

Count Property

Description Returns the number of registered users for the database. This number includes only the users who have created a My Record contact record by previously logging on to the ACT! database. Any users who do not have logon access to the database are not included.

Object Users object

Syntax *object*.**Count**

Value type Short Integer, Read Only

Exists Property

Description Returns True if the specified user exists in the database or False if the user is not in the database. This property verifies whether or not a contact record exists for the user. The user's name is the logon name, not the Contact field value from the corresponding ACT! database record.

Object Users object

Syntax *object*.**Exists** *szName*

Parameters *szName* A string that specifies the user's logon name.

Value type Boolean, Read Only

Name Property

Description Returns the user name of the specified user. The user's name is the logon name, not the Contact field value from the corresponding ACT! database record.

Object Users object

Syntax *object*.**Name** *iUser*

Parameters *iUser* A short integer that specifies the ordinal user number, in the range between 1 and Count.

Value type String, Read Only

See also Count Property

Security Property

Description Gets and sets the security level of the specified user.

Object Users object

Syntax *object*.**Security** *iUser [= lValue*]

Parameters *iUser* A short integer that specifies the ordinal user number, in the range between 1 and Count.

[= lValue] A long integer that specifies the security level for the specified user. Omit this optional parameter to get the security level for the specified user.

Value type Long Integer, Read/Write

Comments This property gets or sets one of the following values:

Value	Description
1	Browse (read-only permission). Can read the records in the database. Cannot add, modify, or delete records.
3	Standard (update, read/write permission on standard functions). Can read the records in the database and can add, delete, and modify records. Cannot add new users to the database, synchronize with another database or user, perform maintenance on the database, or modify the database fields.

Value	Description
5	Administrator (full update and read/write permission). Can perform any database function, including adding new users, synchronizing data, performing database maintenance, and modifying the database fields.

See also Count Property

UniqueId Property

Description Returns the Unique ID of the specified user's contact record in the ACT! database.

Object Users object

Syntax *object*.**UniqueId** *iUser*

Parameters *iUser* A short integer that specifies the ordinal user number, in the range between 1 and Count.

Value type String, Read Only

See also Count Property

Methods

Name	Parameter(s)	Parameter type(s)	Return type
AddUser Method	szName szPassword iPrivilege	String, String Short Integer	Boolean
CheckIsPhonebook Method	None		Boolean
CurrentUserAccess Method	None		Long Integer
CurrentUserId Method	None		String
CurrentUserName Method	None		String
CurrentUserSecurity Method	None		Long Integer
GetPassword Method	iUser	Short Integer	String/BSTR
IsValidPassword Method	iUser, szPassword	Short Integer, String	Boolean
SetAsPhonebook Method	None		Boolean
SetPassword Method	iUser, szOldPassword szNewPassword	Short Integer String String	Void

AddUser Method

Requires ACT! 4.0 or later

Description Adds a new user with a specified security level to an ACT! database. The current user must have Administrator security level to add new users. Returns True if the operation is successful or False if unsuccessful.

Note: Before using this method, ensure the database is not open in the ACT! application or by another instance of OLE.

Object Users object

Syntax *object*.**AddUser** *szName, szPassword, iPrivilege*

Parameters *szName* A string that specifies the new user's logon name.

szPassword A string that specifies the initial password for the new user.

iPrivilege A long integer that specifies the security level for the new user.

Following is a list of values for the iPrivilege parameter:

Value	Description
1	Browse (read-only permission). Can read the records in the database. Cannot add, modify, or delete records.
3	Standard (update, read/write permission on standard functions). Can read the records in the database and can add, delete, and modify records. Cannot add new users to the database, synchronize with another database or user, perform maintenance on the database, or modify the database fields.
5	Administrator (full update and read/write permission). Can perform any database function, including adding new users, synchronizing data, performing database maintenance, and modifying the database fields.

Return type Boolean

See also SetAsMyRecord Method in Contact object

Example

```
' Adds a user to an ACT! database.
Dim objDatabase As Object
Dim objContact As Object
Dim objUsers As Object

Set objDatabase = CreateObject("ACTOLE.DATABASE")
objDatabase.Open("C:\My Documents\ACT\Database\ACT5demo.dbf")
Set objUsers = objDatabase.Users

'Add user "Simon Lazarus" with password "password" and
'security level "1" for browse
if objUsers.AddUser("Simon Lazarus", "password", 1) = True
    then
    MsgBox("Simon Lazarus added successfully");
ElseIf
    MsgBox("Failed adding Simon Lazarus as a database user.");
EndIf

objDatabase.Close
Set objDatabase = Nothing
```

CheckIsPhonebook Method

Requires ACT! 4.0 or later

Description Returns True if the ACT! database is specified in WinFax as an external phonebook or False if it is not specified as a WinFax phonebook.

Note: This method requires WinFax 8.02 or later. False is returned if WinFax 8.02 or later is not installed.

Object Users object

Syntax *object*.**CheckIsPhonebook**

Return type Boolean

See also SetAsPhonebook Method

Example Check if the open database is being used as the WinFax phonebook If not, set it as the phonebook for WinFax.

```
Dim objDatabase As Object
Dim i As Integer
Set objDatabase = CreateObject("ACTOLE.DATABASE")
objDatabase.Open dbName

If objDatabase.IsOpen Then
'If the database is not being used as a WinFax Phonebook,
'Set it to be used as one.
    If objDatabase.USERS.CheckIsPhonebook <> True Then
        i = objDatabase.USERS.SetAsPhonebook
            If i = True Then
                lstDisplay.AddItem "Setting successful"
            Else
                lstDisplay.AddItem "Setting unsuccessful"
            End If
        End If
    objDatabase.Close
End If

'Clear the object
Set objDatabase = Nothing
```

CurrentUserAccess Method

Requires ACT! 3.0.6 or later

Description Returns a long integer containing the access type of the current user of the open database.

Object Users object

Syntax *object.***CurrentUserAccess**

Return type Long Integer

Comments This method returns one of the following values:

Value	Description
1	Logon access to the database is enabled for the selected user. Synchronization access is not enabled for the selected user.
2	Synchronization access to the database is enabled for the specified user. Logon access is not enabled for the selected user.
4	Direct synchronization access to the database has been set up for the selected user.
5	Logon access and synchronization access to the database is enabled for the selected user.

See also CurrentUserId Method, CurrentUserName Method, CurrentUserSecurity Method

CurrentUserId Method

Requires ACT! 3.0.6 or later

Description Returns a string containing the Unique ID of the current user of the open database.

Object Users object

Syntax *object.***CurrentUserId**

Return type String

See also CurrentUserAccess Method, CurrentUserName Method, CurrentUserSecurity Method

CurrentUserName Method

Requires ACT! 3.0.6 or later

Description Returns a string containing the user name of the current user of the open database.

Object Users object

Syntax *object*.**CurrentUserName**

Return type String

See also CurrentUserAccess Method, CurrentUserId Method, CurrentUserSecurity Method

CurrentUserSecurity Method

Requires ACT! 3.0.6 or later

Description Returns a long integer containing the security level of the current user of the open database.

Object Users object

Syntax *object*.**CurrentUserSecurity**

Return type Long Integer

Comments This method returns one of the following values:

Value	Description
1	Browse (read-only permission). Can read the records in the database. Cannot add, modify, or delete records.
3	Standard (update, read/write permission on standard functions). Can read the records in the database and can add, delete, and modify records. Cannot add new users to the database, synchronize with another database or user, perform maintenance on the database, or modify the database fields.
5	Administrator (full update and read/write permission). Can perform any database function, including adding new users, synchronizing data, performing database maintenance, and modifying the database fields.

See also CurrentUserAccess Method, CurrentUserId Method, CurrentUserName Method

GetPassword Method

Requires ACT! 4.0 or later

Description Gets the password of specified user(s). The current user must have an Administrator security level to get user passwords. A password is returned as a string.

Object Users object

Syntax *object*.**GetPassword** *iUser*

Parameters *iUser* A short integer that specifies the ordinal user number, in the range between 1 and Count.

Return type String/BSTR

See also Count Property

Example Example of getting the passwords of users of an ACT! database.

```
Dim objDatabase As Object
Dim objContact As Object
Dim objUsers As Object
Dim uCount As Integer
Set objDatabase = CreateObject("ACTOLE.DATABASE")
objDatabase.Open("C:\My Documents\ACT\Database\ACT5demo.dbf")
Set objUsers = objDatabase.Users
```

```
'Get the count of users.
uCount = objUsers.Count

'Now display the name and password of all users for the database.
for iLoop = 1 to uCount
List1 .Additem "Password for user: " & objUsers.Name(iLoop) & " is: " &
    objUsers.GetPassword(iLoop)
next iLoop

objDatabase.Close
Set objDatabase = Nothing
```

IsValidPassword Method

Description Returns True if the specified password is valid for the specified user or False if the password is invalid or the specified user does not have a password.

Object Users object

Syntax *object*.**IsValidPassword** *iUser, szPassword*

Parameters *iUser* A short integer that specifies the ordinal user number, in the range between 1 and Count.

szPassword A string that specifies the user's current password.

Return type Boolean

See also Count Property

SetAsPhonebook Method

Requires ACT! 4.0 or later

Description Sets an ACT! database as an external phonebook for WinFax. Returns True if the operation is successful or False if unsuccessful.

Note: This method requires WinFax 8.02 or later. False is returned if WinFax 8.02 or later is not installed.

Object Users object

Syntax *object*.**SetAsPhonebook**

Return type Boolean

See also CheckIsPhonebook Method

Example See the example for the CheckIsPhonebook method.

SetPassword Method

Description Sets the password for the specified user. To change the user's password, you must know the user's current password.

Object Users object

Syntax *object*.**SetPassword** *iUser, szOldPassword, szNewPassword*

Parameters *iUser* A short integer that specifies the ordinal user number, in the range between 1 and Count.

*szOldPassword*A string that specifies the user's current password.

*szNewPassword*A string that specifies the user's new password.

Return type Void

See also Count Property, IsValidPassword Method

Chapter 4
The Application Class

This chapter describes the Application class component of the ACT! Software Development Kit (SDK), and lists the properties and methods common to objects derived from it. To use this section, you need to be familiar with and using the following:

- ACT! for Windows, version 3.0.7 or later
- Microsoft Windows 95/98/2000/Me/XP, or Windows NT 4.0.
- Microsoft Visual Basic, version 4.0 or later, or Microsoft Visual C++ 4.0 or later

Microsoft Visual Basic for Applications can be used as an alternative development language to access data in Microsoft Access and Excel versions in Microsoft Office 95, and Microsoft Access, Excel and Word versions in Microsoft Office 97 through Office 2002.

Application class overview

The Application class, also known as the ACT! OLE Application object, provides third-party applications with both control and content integration of ACT!. It is intended for use by both add-on product developers and enterprise product developers who need to tightly integrate ACT! into a larger end-user solution. A comprehensive set of methods and properties is included to control the application frame window, the current view, get or set view grid cells, set application preferences, and more.

Using the Application class, third-party application developers can minimize development time and ensure better compatibility in the future by employing the parts of the ACT! user interface exposed by the OLE Application object to create new ACT! contacts, insert notes or activities, and manage database maintenance operations. Additionally, the Application class allows third-party applications to determine the current context of ACT! (current view, displayed contact or contact group, and so on) as controlled by the user, so that their functionality is more synchronous with the current ACT! UI views.

Rules

When creating an application using the Application class, keep the following in mind:

- ACT! must have been started at least once after ACT! is installed to update the registry entries for the Application class.
- ACT! itself is executed when the object is instantiated but its UI can be hidden through an Application object method if desired. If you hide the ACT! user interface, the ACT! user interface might reappear after an exception condition (such as a confirmation message following the deletion of a record or a lookup failure).
- If a client application starts ACT! or ACT! is running when the client application starts and the user selects EXIT from the File menu, ACT! is hidden, but not closed. ACT! does not close until the client application releases control of ACT! by either closing ACT! or exiting.

- To use the Application object from more than one location in a program, define the Application object as a global variable. This enables the program to access the Application object from many locations and reduces execution time by creating an Application object one time instead of once every time it is needed.
- Some functions within the Application object set the focus to ACT! to keep ACT! as the foreground application. ACT! requires the focus to perform actions such as receiving Windows messages or using data stored in lists inside the application. Code needs to be added to a third-party application to bring it to the foreground as soon as it is done interacting with the OLE Application object.
- Where possible, descriptions include error codes (in the heading GetLastError) that can be returned. Use the GetLastError Method (described in Common properties and methods) to get the last error code for the object.
- Methods with a long integer or short integer return type return S_OK on success.
- All objects should be declared and created before use and closed and set to nothing afterwards, implicitly (as in a macro) or explicitly. Using close methods before setting objects to null ensures all memory is released and significantly improves performance. Examples in this document illustrate concepts only, and do not always contain all of these steps.

System requirements

The following applications must be already installed on the SDK computer:

- ACT! for Windows, version 3.0.7 or later. Interact Commerce Corporation recommends using the latest update. To receive an update, choose ACT! Update from the Help menu.
- Microsoft Windows 95/98/2000/Me/XP or Windows NT 4.0

Using the Database class with Visual C++

Many programmers who work with the OLE Application object work with Microsoft Visual Basic. Hence, most of the examples in the documentation of the application object are designed for the Visual Basic programmer. However, a type library is supplied and consequently, the integrated development environment (IDE) of Microsoft Visual C++ can be used to automatically generate prototypes for the object methods.

The following basic steps can help getting started using ACT! automation libraries using Microsoft Visual C++ 4.0 or later and Microsoft Foundation Class (MFC) library. To use the Application class in an application written using Visual C++, first create a wrapper class around the OLE objects supported by the OLE Application class.

To create a wrapper class

1. Create a new MFC project and enable OLE automation.
2. Open the class wizard and click **Add Class**.
3. Select the **From Type Library** option.
4. When prompted for the name of the type-library, choose ACT.TLB from the ACT! program files folder.
5. In the **Confirm Classes** dialog box, click **OK** or select only those classes that you need, then click **OK**.

Visual C++ automatically generates MFC wrapper classes for all methods and properties supported by the OLE Application object. For more information on how to use OLE automation in Visual C++, see Visual C++ online Help.

For each property, the ACT! Application object creates corresponding sets of Get and Set methods in an MFC wrapper class. Read-only properties, however, have only a Get method. If using Visual C++, check the C++ header file generated by Visual C++ (ACT.H) for the methods that correspond to properties documented in this manual. The following table lists examples of properties and corresponding methods to use in Visual C++.

Property name	Type	C++ Method
ContactView.Active	Read Only	ContactView.GetActive()
Preferences.ExitPrompt	Read/Write	Preferences.GetExitPrompt() - Gets the value of the property. Preferences.SetExitPrompt() - Gets the value of the property.

Sample code

```
//This example lists the current contact's name, address and phone number
IActAppObj AppObj;
IAIViews ViewsObj;
IAIContactView ContactViewObj;

//Create a Dispatch pointer to the application object
AppObj.CreateDispatch("ACTOLE.APPOBJECT", pException);

//Set your own caption
AppObj.SetCaption("My Extended ACT!");
m_ListBox.ResetContent()
if (AppObj.IsDBOpen ())//If a database is open
{
    //Attach the Views object
    LPDISPATCH viewsDispatch = AppObj.Views();
    ViewsObj.AttachDispatch(viewsDispatch, TRUE);

    //Attach the ContactView object
    LPDISPATCH CVDispatch = ViewsObj.Create(1,"CView");
        ContactViewObj.AttachDispatch(CVDispatch, TRUE);

    //List information about the current contact
    m_ListBox.AddString("Current Contact Information:");
    m_ListBox.AddString(ContactViewObj.GetField(CF_Name) );
    m_ListBox.AddString(ContactViewObj.GetField(CF_Title) );
    m_ListBox.AddString(ContactViewObj.GetField(CF_Company) );
    m_ListBox.AddString(ContactViewObj.GetField(CF_Phone) );
    m_ListBox.AddString(ContactViewObj.GetField(CF_Address1) );
    m_ListBox.AddString(ContactViewObj.GetField(CF_City) );
    m_ListBox.AddString(ContactViewObj.GetField(CF_State) + " "
        + ContactViewObj.GetField(CF_Zip) );
    m_ListBox.AddString(ContactViewObj.GetField(CF_Zip) );

    //Release all Dispatch pointers
        ContactViewObj.ReleaseDispatch();
    ViewsObj.ReleaseDispatch();
    AppObj.ReleaseDispatch();
}
```

Understanding key files and concepts

The following table lists key ACT! files used by the ACT! OLE Database object. These files are stored in the ACT! program files folder.

File	Description
ACT.EXE	The ACT! Application object is contained within the executable file for the ACT! application.
ACT.TLB	Type library that contains functions within the ACT! OLE Application object. Methods contained in the type library need to be used directly by Visual C++ developers. In Visual Basic type library functions are handled automatically at run time.
ACTEVENT.OCX	OLE event notification control module, used by third-party applications for receiving event notification. An event is generated, for example, when the contact is changed in the Contact View.
ACTREG.EXE	The ACT! Windows registry update utility. This utility can be run manually if necessary to register ACT! OLE controls as a troubleshooting procedure.

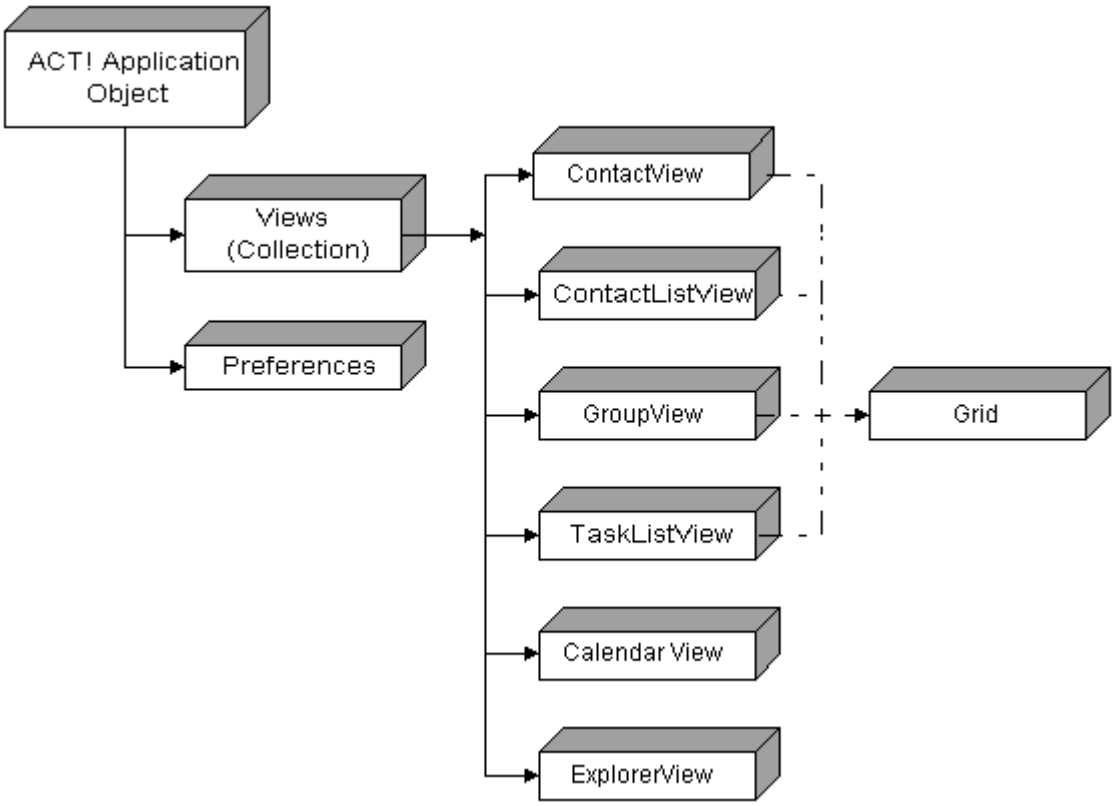
Object definitions

The Application class has the following derived objects:

Object	Definition
Application object	Is the OLE creatable object that provides functions to externally control the ACT! application.
CalendarView object	Provides an interface to the Calendar view functionality.
ContactListView object	Provides an interface to the Contact List view functionality.
ContactView object	Provides an interface to the Contact view functionality.
ExplorerView object	Provides an interface to functionality within floating views and tabs added using control files.
Grid object	Provides an interface to all ACT! grid functionality, including the Contact List View, the Task List view, and the contact tabs Activities, Groups, Notes/Histories, and Sales. Use the Grid object to retrieve data from any of the listed views.
GroupView object	Provides an interface to the Contact view functionality.
Preferences object	Allows the SDK to manipulate user preferences in the ACT! application.
TaskListView object	Provides an interface to the Task List view functionality.
Views object	Provides an interface to manage all ACT! views.

Application object model

The following chart shows the relationships of objects in the ACT! OLE application object model:



Error codes

The following error codes apply only to the ACT! application. If an error occurs that is not caused by the OLE automation client's code, please record the return value and the situation that caused it for use by technical support.

Value	Error code	Description
-1		General error
-2		Row is not visible in window
-10	S_UNKNOWN	Unknown internal error
100	S_ERROR	General error
101	S_NONE	No more items
102	S_NOT_FOUND	File not found
103	S_MAIN_WND	Application window not found
104	S_FRM_VW_SYS	Internal error
105	S_CON_DOC	No open database
106	S_GRP_FRM	Internal error
107	S_JUMP_TO_REC	Internal error
108	S_LOOKUP_FAIL	Internal error

Value	Error code	Description
109	S_ACTIVE_REC	Internal error
110	S_ACTIVE_TAB	Not used
111	S_ACTIVATE_TAB	Not used
112	S_GETFIELD_FAIL	Internal error
113	S_SETFIELD_FAIL	Internal error
114	S_INTERNAL	Not used
115	S_LIST_ERR	Internal error
116	S_VW_NOT_FOUND	Specified view does not exist
117	S_INVALID_INPUT	Invalid input
118	S_NO_COMPOSE_VW	Mail Compose view does not exist
119	S_GET_SELECTION_FAIL	Internal error
120	S_FOLDER_OPEN_FAIL	Internal error
121	S_VW_CREATE_FAIL	Not used
122	S_PROPS_NOT_FOUND	Preferences not found (internal error)
123	S_WRITE_FAIL	Failed to change user preferences (internal error)
124	S_NO_WP_DRVMGR	Internal error
125	S_NO_WP_DRIVER	No word processor driver (internal error)
126	S_NO_FAX_DRVMGR	Internal error
127	S_NO_FAX_DRIVER	No fax driver (internal error)
128	S_NO_DB	No open database
129	S_NO_SECURITY	Internal error
130	S_NOT_PRIVILEGED	Not enough privileges to perform operation
131	S_CREATE_NEWUSER	Internal error
132	S_NO_USER	No current user
133	S_NO_MATCH	Old and new passwords do not match
134	S_MEM_ERROR	Error allocating memory
135	S_NOMENU	Internal error
136	S_INVALID_CMD	Invalid command
137	S_CON_LIST	Internal error
138	S_FILE_OPEN	Unable to open specified file
139	S_MACRO_ERROR	Error running specified macro (internal error)
140	S_NO_CONTAINER	Not used
141	S_NO_DALLIST_VIEW	Internal error
142	S_VIEW_EXISTS	A view of the specified type already exists
143	S_NO_EMAIL	E-mail view does not exist
144	S_NO_CALVW	Calendar view does not exist
145	S_INVALID_VWTYPE	View type is invalid
146	S_NO_OPENDB	No open database

Value	Error code	Description
147	S_NO_TAB_CONTAINER	Internal error
148	S_DEFAULT_USED	Input was invalid. Default was used. (SetSynchSettings method)
148	S_INVALID_TAB	Invalid tab specified (SetActiveTab method)
149	S_ACTIVITY_INIT_FAIL	Internal error
150	S_HELPER_EEDIT_FAIL	Internal error
151	S_ADD_CONTACT	Internal error
152	S_NO_RECORD	No active record
153	S_RECORD_HISTORY	Could not record history
154	S_DUPLICATE_CONTACT	Duplicate contact
155	S_DUPLICATE_GRP	Duplicate group
156	S_DELETE_FAIL	Delete failed (internal error)
157	S_DUPLICATE	Internal error
158	S_BUILD_ERROR	Error building list (internal error)
159	S_CANT_SORT	Not used
160	S_STOP_EDIT	Internal error
161	S_NO_SCHEMA	Internal error
162	S_NOT_SORTABLE	Specified field is not sortable
163	S_INVALID_ID	Invalid ID specified
164	S_SET_NOT_ALLOWED	Set not allowed on specified field
165	S_DELETE_NOTALLOWED	Row cannot be deleted from the specified grid
166	S_NOT_IMPLEMENTED	Method not implemented
167	S_NO_OVERWRITE	File could not be overwritten
168	S_ASK_OVRWRITE	Not used
169	S_NOT_READY	Destination device is not ready
170	S_OPEN_FILE	Not used
171	S_ADD_FILE	Not used
172	S_SERVER_BUSY	ACT! application is waiting for user response
173	S_FILE_EXISTS	Destination file exists
174	S_WRITE_PROTECT	Destination file is write protected
175	S_NO_EMAIL_SYSTEM	E-mail system is not setup
176	S_NOT_SUPPORTED	Method is not supported
177	S_NO_OUTLOOK	No Outlook on the system
178	S_BAD_PARAM	Invalid parameter values
179	S_GRP_HAS_SUBGRP	The group has a subgroup (Delete method in GroupView object))
180	S_BAD_FILENAME	Invalid pathname, filename, or extension.
181	S_SCRIPT_ERROR	Script file for the TriggerActivitySeries method encountered an error.

Common properties and methods

The following common properties and methods all apply to the CalendarView object, ContactListView object, ContactView object, ExplorerView object, GroupView object, TaskListView object unless otherwise noted.

Properties

Name	Parameter(s)	Parameter type(s)	Value type	Access
Active Property	None	*	Boolean	Read Only
Caption Property	[=szTitleBar]	String/BSTR	String/BSTR	Read/Write
Displayed Property	None	*	Boolean	Read Only
Name Property	[=szName]	String/BSTR	String/BSTR	Read/Write
Type Property	None	*	Long Integer	Read Only
ViewState Property	None	*	Short Integer	Read Only

Active Property

Description Returns True if the view is active or False if it is inactive or an error occurs. A view can be visible but inactive.

Objects CalendarView object, ContactListView object, ContactView object, ExplorerView object, GroupView object, TaskListView object

Syntax *object*.**Active**

Value type Boolean, Read Only

GetLastError S_VW_NOT_FOUND

See also Activate Method, Displayed Property, Show Method

Example See Activate method.

Caption Property

Description Gets and sets the text in the title bar for the view.

Objects CalendarView object, ContactListView object, ContactView object, ExplorerView object, GroupView object, TaskListView object

Syntax *object*.**Caption** [= *szTitleBar]*

Parameters *szTitleBar* A string that specifies the text in the title bar for the view. Omit this optional parameter to get the text in the title bar.

Value type String/BSTR, Read/Write

GetLastError S_VW_NOT_FOUND

Example

```
'This example gives the Task List window a new title bar.
Dim objApp as object
Dim objViews as object
Dim objTaskView as object

'Initialize the Application object.
'This starts ACT! if it is not already running.
Set objApp = CreateObject("ACTOLE.APPOBJECT")
Set objViews = objApp.Views
Set objTaskView = objViews.Create (4,"TL")
```

```
'Change the text on the Task List title bar to "My Task View".
objTaskView.Caption = "My Task View"

'Close the Application object.
Set objApp = Nothing
```

Displayed Property

Description Returns True if the view is visible or False if the view is not visible.

Objects CalendarView object, ContactListView object, ContactView object, ExplorerView object, GroupView object, TaskListView object

Syntax *object*.**Displayed**

Value type Boolean, Read Only

GetLastError S_VW_NOT_FOUND

See also Activate Method, Active Property, Show Method

Example See the Activate method.

Name Property

Description Sets or gets the name of the view.

Objects CalendarView object, ContactListView object, ContactView object, GroupView object, TaskListView object

Syntax *object*.**Name**[=*szName*]

szName A string that specifies the name of the view. Omit this optional parameter to get the name of the view.

Value type String/BSTR, Read/Write

See also Caption Property

Type Property

Description Returns the type of view. The property is set during creation of the view object using the Create method in the Views object.

Objects CalendarView object, ContactListView object, ContactView object, ExplorerView object, GroupView object, TaskListView object

Syntax *object*.**Type**

Value type Long Integer, Read Only

Comments This property returns one of the following values:

Value	State	Value	State
1	Contact view	5	Calendar view
2	Contact List view	6	E-mail inbox view
3	Group view	7	Explorer view
4	Task List view		

Example See the GetActive Method in the Views object.

ViewState Property

Requires ACT! 5.0 or later

Description Returns the current view's display state.

Object CalendarView object, ContactListView object, ContactView object, ExplorerView object, GroupView object, TaskListView object

Syntax *object*.**ViewState**

Value type Short Integer, Read Only

Comments This property returns one of the following values:

Value	State	Value	State
1	Normal	3	Minimized
2	Maximized	4	Hidden

See also CreateEx Method in Views object

Methods

Name	Parameter(s)	Parameter type(s)	Return type
Activate Method	None	*	Boolean
Application Method	None	*	Object/LPDISPATCH
ClearError Method	None	*	Void
Close Method	None	*	Long Integer
CurrentFieldId Method	None	*	Long Integer
CurrentRecord Method	None	*	Long Integer
GetLastError Method	None	*	Long Integer
GetMode Method	None	*	Short Integer
HasRecordChanged Method	None	*	Boolean
LookupKeyword Method	szKeyword lTables LookupType szGroupName	String, Long Integer Short Integer String	Long Integer
Maximize Method	None	*	Boolean
Minimize Method	None	*	Boolean
Parent Method	None	*	Object/LPDISPATCH
ReSize Method	iHeight iWidth	Short Integer Short Integer	Long Integer
Show Method	True|False	Boolean	Long Integer
Update Method	None	*	Long Integer

Activate Method

Description Activates the view by bringing it to the foreground. Returns True on success or False on failure.

Objects CalendarView object, ContactListView object, ContactView object, ExplorerView object, GroupView object, TaskListView object

Syntax *object.***Activate**

Return type Boolean

GetLastError S_VW_NOT_FOUND

See also Active Property, Displayed Property

Example

```
'This example can be put into a function that has been passed only
'the Views object - objViews. We assume in the main function that
'the GroupView object has already been created.

Dim objGrp1 as object

'Get the GroupView pointer.
Set objGrp1 = objViews.GetView(3)

'If the Group View is not visible, then make it visible.
If objGrp1.Displayed = False then
   objGrp1.Show True
End If

'If the GroupView is not the active view, then make it the active view.
If objGrp1.Active = False then
   objGrp1.Activate
End If

'Now any GroupView methods can be applied to manipulate the Group view.

'End the function
Set objGrp1 = Nothing
```

Application Method

Description Returns a dispatch pointer to the Application object containing the view. The returned pointer can be used to control the Application object.

Objects CalendarView object, ContactListView object, ContactView object, ExplorerView object, GroupView object, TaskListView object

Syntax *object.***Application**

Return type Object/LPDISPATCH

ClearError Method

Requires ACT! 4.0 or later

Description Clears the last error.

Objects CalendarView object, ContactListView object, ContactView object, ExplorerView object, GroupView object, TaskListView object

Syntax *object.***ClearError**

Return type Void

See also GetLastError Method

Close Method

Description Closes the view. Returns S_ERROR on failure.

Objects CalendarView object, ContactListView object, ContactView object, ExplorerView object, GroupView object, TaskListView object

Syntax *object*.**Close**

Return type Long Integer

GetLastError S_VW_NOT_FOUND

CurrentFieldId Method

Requires ACT! 5.0 or later

Description Returns the field ID of the field containing the cursor. See ACT! Databases for lists of field IDs and names (Field constants).

Objects ContactView object, GroupView object

Syntax *object*.**CurrentField**

Return type Long Integer

CurrentRecord Method

Requires ACT! 5.0 or later

Description Returns the current Contact or Group record number and position for the lookup.

Objects ContactView object, GroupView object

Syntax *object*.**CurrentRecord**

Return type Long Integer

Example

```
'The following sample enumerates all the contacts and their record numbers.

Set objViews = objApp.Views
Set objContactView = objViews.Create(1, "CL")
objContactView.MoveFirst
'Move through all the contacts.
For i = 1 To objContactView.GetCount
    'Display the record number and the Contact name.
    lstVerify.AddItem "Contact # " & objContactView.CurrentRecord &
        " is : " & objContactView.GetField(CF_Name)
    objContactView.MoveNext
Next i
```

Example

```
'The following code enumerates the group and subgroup names and their
'record number positions in the current lookup.
'If a group tree is collapsed, it is expanded.

Set objViews = objApp.Views
Set objGrp = objViews.CreateEx(3, "GV",2)
objGrp.MoveFirst
'Go through all the groups.
For i = 0 To objGrp.GetCount - 1
    'If the group tree is not expanded, expand it.
    If objGrp.IsExpanded = False Then
        objGrp.Expand
    End If
```

```
    'List the record number and group name.
    List1.AddItem objGrp.CurrentRecord & " ---" & objGrp.GetField(GF_Name)
    objGrp.MoveNext
Next i
```

GetLastError Method

Description Returns a long integer representing the last error code for the object. For a list of error codes, see Error codes.

Objects CalendarView object, ContactListView object, ContactView object, ExplorerView object, GroupView object, TaskListView object

Syntax *object*.**GetLastError**

Return type Long Integer

See also ClearError Method

GetMode Method

Requires ACT! 5.0 or later

Description Returns a short integer identifying the task that the Contact or Group form is currently used for. This method determines if the form is in data entry/browse mode, lookup by example mode, or replace fields mode. Operations on the view should be performed in data entry/browse mode.

Objects ContactView object, GroupView object

Syntax *object*.**GetMode**

Return type Short Integer

Comments The following values are returned by this method:

Value	Mode	Value	Mode
0	None (for example, when a dialog box is open)	3	Query
1	Data Entry/Browse	4	Replace Fields
2	Layout		

Example

```
'The following example verifies that the Contact view is in data entry/
'browse mode before enumerating the contacts in the database.

Set objContactView = objViews.CreateEx(1, "CV", 1)
'If the mode is not 1 (not Data Entry/Browse)
If objContactView.GetMode <> 1 Then
    Select Case objContactView.GetMode
    Case 0
        MsgBox "A dialog is open in ACT!, please close it and try again"
    Case 2
        MsgBox "The Layout Editor is open in ACT!, please close it and try
               again"
    Case 3
        MsgBox "The Lookup by Example/Query is open in ACT!, please close it
               and try again"
    Case 4
        MsgBox "Replace Fields is open in ACT!, please close it and try
  again"
    End Select
```

```
Else
    'Now enumerate all the contacts in the database.
    objContactView.MoveFirst
    For i = 1 To objContactView.GetCount
        lstVerify.AddItem objContactView.GetField(CF_Name)
        objContactView.MoveNext
    Next i
End If
```

HasRecordChanged Method

Requires ACT! 5.0 or later

Description Returns True if the user has changed the current Contact or Group record (without saving the record) or False if the current record has not been changed.

Objects ContactView object, GroupView object

Syntax *object*.**HasRecordChanged**

Return type Boolean

LookupKeyword Method

Requires ACT! 5.0 or later

Description Looks up the keyword in the specified tables and displays the Lookup Keyword dialog containing the results.

Object ContactView object, GroupView object

Syntax *object*.**LookupKeyword** (*szKeyword*, *lTables*, *iLookupType*, *szGroupName*)

Parameters *szKeyword* A string specifying the keyword.

lTables A long integer specifying the tables to be searched. This value is created by ORing together values that represent the tables.

Following is a list of the values for the lTables parameter:

Value	Table	Value	Table
1	Contact	8	Email
2	Activity	16	Groups
4	NoteHistory	32	Sales

iLookupType A short integer specifying the lookup type.

Following is a list of the values for the iLookupType parameter:

Value	Type	Value	Type
0	All Records	2	Current Lookup
1	Current Record	3	Selected Group

*szGroupName*If the lookup type is 3, specify a string containing the Unique ID of the selected group, otherwise specify a blank group name.

Return type Long Integer

Comments Returns S_OK or S_ERROR.

Example

```
Set objViews = objApp.Views
Set objContactView = objViews.Create(1, "CL")

'Lookup keyword "SDK" in the Contact view in the Contact, Activities and
'Notes/History fields for contacts belonging to a group with Unique ID GUID
objContactView.LookupKeyword "SDK", 1 Or 2 Or 4, 3, GUID
```

Maximize Method

Description Maximizes the view. Returns True on success or False on failure.

Objects CalendarView object, ContactListView object, ContactView object, ExplorerView object, GroupView object, TaskListView object

Syntax *object*.**Maximize**

Return type Boolean

GetLastError S_VW_NOT_FOUND

See also Minimize Method, Show Method

Minimize Method

Description Minimizes the view. Returns True on success or False on failure.

Objects CalendarView object, ContactListView object, ContactView object, ExplorerView object, GroupView object, TaskListView object

Syntax *object*.**Minimize**

Return type Boolean

GetLastError S_VW_NOT_FOUND

See also Maximize Method, Show Method

Parent Method

Description Returns a dispatch pointer to the Views object that contains this view. The returned pointer can be used to control the Views object.

Objects CalendarView object, ContactListView object, ContactView object, ExplorerView object, GroupView object, TaskListView object

Syntax *object*.**Parent**

Return type Object/LPDISPATCH

ReSize Method

Description Sets the size of the view. The current position (Top Left) of the view remains unchanged. Returns S_ERROR on failure.

Objects CalendarView object, ContactListView object, ContactView object, ExplorerView object, GroupView object, TaskListView object

Syntax *object*.**ReSize**(*iHeight*, *iWidth*)

Parameters

iHeight	A short integer indicating the new height of the view.
iWidth	A short integer indicating the new width of the view.

Return type Long Integer

GetLastError S_VW_NOT_FOUND

See also Maximize Method, Minimize Method, Show Method

GetSize Method in the Application object

Example

```
'This example resizes the view.
Dim objApp as object

'Initialize the Application object.
'This starts ACT! if it is not already running.
Set objApp = CreateObject("ACTOLE.APPOBJECT")

'Resize the view.
objApp.ReSize 500, 800

'Close the Application object.
Set objApp = Nothing
```

Show Method

Description Hides or displays the view. When used on a maximized view, all views are set to normal size to enable selection of the new active view. Once a new active view has been selected, views are maximized again, and hidden views display behind active one.

Objects CalendarView object, ContactListView object, ContactView object, ExplorerView object, GroupView object, TaskListView object

Syntax *object*.Show(*True*|*False*)

Parameters *True*|*False* Specify True to display the frame or False to hide it.

Return type Long Integer

GetLastError S_VW_NOT_FOUND

See also Activate Method, Maximize Method, Minimize Method

Example

```
'This example displays the application window.
Dim objApp as object

'Initialize the Application object.
'This starts ACT! if it is not already running.
Set objApp = CreateObject("ACTOLE.APPOBJECT")

'Open the database.
objApp.OpenFile C:\My Documents\ACT\Database\ACT5demo.dbf

'If the application is not visible, then display it.
If objApp.IsVisible = False then
   objApp.Show (True)
End If

'Close the Application object.
Set objApp = Nothing
```

Update Method

Description Redraws the view. Returns S_ERROR on failure.

Objects CalendarView object, ContactListView object, ContactView object, ExplorerView object, GroupView object, TaskListView object

Syntax *object*.**Update**

Return type Long Integer

GetLastError S_VW_NOT_FOUND

Chapter 5
Objects Derived from the Application Class

This chapter discusses the objects derived from the Database class and their associated methods and properties. Objects include:

Name	Name
Application object	Grid object
CalendarView object	GroupView object
ContactListView object	Preferences object
ContactView object	TaskListView object
ExplorerView object	Views object

Application object

The Application object provides functions to externally control the ACT! application. The following properties and methods apply only to the Application object. See Common properties and methods for properties and additional methods that apply to this object.

Properties

Name	Parameter(s)	Parameter type(s)	Value type	Access
ActVersion Property	None	*	String	Read Only
Caption Property	[= szTitleBar]	String/BSTR	String/BSTR	Read/Write
LastContactListModTime Property	None	*	Long Integer	Read Only

ActVersion Property

Requires ACT! 5.0 or later

Description Returns a string that contains the version of the ACT! application. An example of a returned string is 5.0.0.175, where 5.0.0 is the version of ACT! (ACT! 5.0) and 175 is the number of the build.

Object Application object

Syntax *object*.**ActVersion**

Value type String, Read Only

Caption Property

Description Gets and sets the text in the title bar for the view.

Object Application object

Syntax *object*.**Caption** [= *szTitleBar*]

Parameters *szTitleBar* A string that specifies the text to appear in the title bar for the view. Omit this optional parameter to get the text that appears in the title bar.

Value type String/BSTR, Read/Write

GetLastError S_MAIN_WND

Example

```
'This example gives the ACT! application a new caption.
Dim objApp as object

'Initialize the Application object.
'This starts ACT! if it is not already running.
Set objApp = CreateObject("ACTOLE.APPOBJECT")

'Change the ACT! title bar from "ACT!" to "My ACT! Application"
App.Caption = "My ACT! Application"

'Close the Application object.
Set objApp = Nothing
```

LastContactListModTime Property

Description Returns the time at which the current lookup was last modified.

Object Application object

Syntax *object*.**LastContactListModTime**

Value type Long Integer, Read Only

Methods

Name	Parameter(s)	Parameter type(s)	Return type
AddUser Method	szUserName szPassword iPrivilege	String String Short Integer	Long Integer
BackupDB Method	szZipFileFullPath iItems iMode	String Short Integer Short Integer	Long Integer
ChangePassword Method	szUserName szOldPassword szNewPassword	String String String	Long Integer
ClearError Method	None	*	Void
CloseDB Method	None	*	Long Integer
Command Method	iCommandID	Short Integer	Long Integer
CompressDB Method	szDBName	String	Long Integer
GetAppName Method	None	*	String/BSTR
GetAppPath Method	None	*	String/BSTR
GetCurrentUserName Method	None	*	String/BSTR
GetLastError Method	None	*	Long Integer
GetOpenDBName Method	None	*	String/BSTR
GetPosition Method	lXPos lYPos	Long Integer Long Integer	Long Integer

Name	Parameter(s)	Parameter type(s)	Return type
GetSize Method	lWidth lHeight	Long Integer Long Integer	Long Integer
GetUserId Method	None	*	String/BSTR
GetUserPrivilege Method	None	*	Long Integer
GetVersion Method	iFor	Short Integer	String/BSTR
Help Method	[szHelpFile lContextID]	String Long Integer	Long Integer
IsDBOpen Method	None	*	Boolean
IsVisible Method	None	*	Boolean
Maximize Method	None	*	Long Integer
Minimize Method	None	*	Long Integer
OpenDB Method	szDBName	String	Object/LPDISPATCH
OpenFile Method	szDBName	String	Long Integer
Preferences Method	None	*	Object/LPDISPATCH
ProcessFile Method	szFilename lReserved	String Long Integer	Long Integer
PurgeHistories Method	szDBName	String	Long Integer
PurgeNotes Method	szDBName	String	Long Integer
PurgeTransactions Method	szDBName	String	Long Integer
ReIndexDB Method	szDBName	String	Long Integer
RemoveOutlookActivities Method	iRemoveIn	Short Integer	Long Integer
ReSize Method	iHeight,iWidth	Short Integer,Short Integer	Long Integer
RestoreDB Method	szSourceZipFile szDestinationPath iMode	String String Short Integer	Long Integer
RunMacro Method	szMacroName	String	Long Integer
SaveCurrentLookup Method	szFilename	String	Long Integer
SendKey Method	lParam1 lParam2 lParam3	Long Integer Long Integer Long Integer	Long Integer
Show Method	True|False	Boolean	Long Integer
Update Method	None	*	Long Integer
UpdateOutlookActivities Method	iUpdateDirection, iDurationType	Short Integer, Short Integer	Long Integer
Views Method	None	*	Object/LPDISPATCH

AddUser Method

Description

Adds a user with the specified password and security level to the open database, but does not verify that a My Record exists for the user. The database should not be open by the ACT! application or by another instance of OLE. The current user must have Administrator security level to add new users. Returns S_ERROR on failure.

Note: This method is not supported in C++, so load the Users object and call the AddUser Method specific to that object instead.

Object Application object

Syntax *object*.**AddUser** (*szUserName, szPassword, iPrivilege*)

Parameters *szUserName* A string representing the user name of the new user.

szPassword A string representing the password for the new user.

iPrivilege A short integer representing the privilege (security level) of the new user.

Following is a list of the values for the iPrivilege parameter:

Value	Security level	Value	Security level
0	Browse	2	Administrator
1	Standard		

Return type Long Integer

GetLastError S_NO_DB, S_NO_SECURITY, S_NO_USER, S_NOT_PRIVILEGED, S_CREATE_NEWUSER

See also ChangePassword Method

Example

```
'This example adds the user "Tim Kelly".
Dim objApp as object

'Initialize the Application object.
'This starts ACT! if it is not already running.
Set objApp = CreateObject("ACTOLE.APPOBJECT")

'Add the user "Tim Kelly" with a password "User"
'with privilege to browse only.
objApp.AddUser "Tim Kelly", "User", 0

'Close the Application object.
Set objApp = Nothing
```

BackupDB Method

Requires ACT! 4.0 or later

Description Backs up the current database to the specified zip file name and location. Returns S_ERROR on failure.

Object Application object

Syntax *object*.**BackupDB** (*szZipFileFullPath, iItems, iMode*)

Parameters *szZipFileFullPath*A string representing the name and path of the zip file to contain the backed up database. The default file extension of .ZIP is used if an extension is not specified.

iItems A short integer that specifies the items to be included in the backup. This value is created by ORing together values that represent the items to be included in the backup.

The following table lists the values for items to be included in the backup:

Value	Item	Value	Item
1	Reports	4	Envelopes
2	Labels	8	Layouts

iMode Specify 168 to overwrite existing backup files located as specified in the szZipFileFullPath parameter. Specify 167 to prevent overwriting existing backup files and return without creating a backup if existing backup files are found in the specified location.

Return type Long integer

See also RestoreDB Method

ChangePassword Method

Description Changes the password for the specified user name. Returns S_ERROR on failure.

Object Application object

Syntax *object*.**ChangePassword** (*szUserName, szOldPassword, szNewPassword*)

Parameters

szUserName A string representing an existing user name.

szOldPassword A string representing the current password.

szNewPassword A string representing the new password.

Return type Long Integer

Comments To be able to change the password, the current user must have Administrator privilege or szUserName must be the current user. If szOldPassword and the currently registered password for szUserName do not match, the method fails.

GetLastError S_NO_DB, S_NO_SECURITY, S_NO_USER, S_NOT_PRIVILEGED, S_NO_MATCH

See also AddUser Method

Example

```
'This example changes the password for an existing user.
Dim objApp as object

'Initialize the Application object.
'This starts ACT! if it is not already running.
Set objApp = CreateObject("ACTOLE.APPOBJECT")

'Change the password of user Tim Kelly from "User" to "Password".
objApp.ChangePassword "Tim Kelly", "User", "Password"

'Close the Application object.
Set objApp = Nothing
```

ClearError Method

Requires ACT! 4.0 or later

Description Clears the last error.

Object Application object

Syntax *object*.**ClearError**

Return type Void

See also GetLastError Method

CloseDB Method

Description Closes the database and all open views. Returns S_ERROR on failure.

Object Application object

Syntax *object*.**CloseDB**

Return type Long Integer

GetLastError S_ERROR

Command Method

Description Executes the specified command. Returns S_ERROR on failure or if the specified command ID is not valid in the current context.

Object Application object

Syntax *object*.**Command** (*iCommandID*)

Parameters *iCommandID* A short integer representing the command ID. For a list of values for this parameter, see Command IDs.

Return type Long Integer

GetLastError S_NOMENU, S_INVALID_CMD

CompressDB Method

Description Compresses the specified database. The database to be compressed should not be open. Use this method to maintain the database. Returns S_ERROR on failure.

Object Application object

Syntax *object*.**CompressDB** (*szDBName*)

Parameters *szDBName* A string representing the name of the database that is to be compressed.

Return type Long Integer

GetLastError S_NO_DB, S_INVALID_INPUT

Comments It is possible that some database maintenance operations are occurring in the background after this method returns. Other database maintenance methods should not be invoked immediately after CompressDB.

See also PurgeHistories Method, PurgeNotes Method, PurgeTransactions Method, ReIndexDB Method

Example **Note**: Another database must be open before you can call these methods.

```
'This example reindexes and compresses the database "CONTACTS.DBF".
Dim objApp as object

'Initialize the Application object.
'This starts ACT! if it is not already running.
Set objApp = CreateObject("ACTOLE.APPOBJECT")

'Open ACT5DEMO.DBF.
objApp.OpenFile C:\My Documents\ACT\Database\ACT5demo.dbf

'Reindex and compress the database CONTACTS.DBF, which is not open.
objApp.ReIndexDB C:\My Documents\ACT\Database\CONTACTS.dbf
objApp.CompressDB C:\My Documents\ACT\Database\CONTACTS.dbf

'Close the Application object.
Set objApp = Nothing
```

GetAppName Method

Description Gets the name of the application. Returns NULL on failure.

Object Application object

Syntax *object*.**GetAppName**

Return type String/BSTR

See also Caption Property, GetAppPath Method, GetOpenDBName Method

Example

```
'This example lists the application name and path in a list box.
Dim objApp as object

'Initialize the Application object.
'This starts ACT! if it is not already running.
Set objApp = CreateObject("ACTOLE.APPOBJECT")

'List the application name in a list box. If the ACT! folder
'has an ACT.EXE this should list "ACT"
List1.AddItem objApp.GetAppName

'If ACT.EXE is in the C:\PROGRAM FILES\ACT folder,
'then "C:\PROGRAM FILES\ACT" should be listed.
List1.AddItem objApp.GetAppPath

'Close the Application object.
Set objApp = Nothing
```

GetAppPath Method

Description Gets the path from where the ACT! application is currently executing. Returns NULL on failure.

Object Application object

Syntax *object*.**GetAppPath**

Return type String/BSTR

GetLastError S_ERROR

See also GetAppName Method

Example See GetAppName.

GetCurrentUserName Method

Requires ACT! 4.0 or later

Description Returns a string containing the user name of the user currently logged on to the currently open database in the ACT! application.

Object Application object

Syntax *object*.**GetCurrentUserName**

Return type String/BSTR

GetLastError Method

Description Returns a long integer representing the last error code for the object. For a list of error codes, see "Error codes".

Object Application object

Syntax *object*.**GetLastError**

Return type Long Integer

GetOpenDBName Method

Description Gets the file name of the active database. Returns NULL on failure.

Object Application object

Syntax *object*.**GetOpenDBName**

Return type String/BSTR

GetLastError S_CON_DOC

See also GetAppName Method

Example

```
'This example gets the file name of the active database.
Dim objApp as object

'Initialize the Application object.
'This starts ACT! if it is not already running.
Set objApp = CreateObject("ACTOLE.APPOBJECT")

'If a database is already open.
If objApp.IsDBOpen = True then
    'If the open database is not ACT5DEMO.DBF, then open it.
    If objApp.GetOpenDBName <>
("C:\My Documents\ACT\Database\ACT5demo.dbf") then
        objApp.OpenDB C:\My Documents\ACT\Database\ACT5demo.dbf
    End If
End If
Set objDatabase = Nothing

'Close the Application object.
Set objApp = Nothing
```

GetPosition Method

Description Retrieves the position of the application's frame window.

Object Application object

Syntax *object*.**GetPosition** (*lXPos, lYPos*)

Parameters *lXPos* A long integer (pointer in Visual C++) representing the X coordinate of the window's top-left position.

lYPos A long integer (pointer in Visual C++) representing the Y coordinate of the window's top-left position.

Return type Long Integer

See also GetSize Method, ReSize Method

Example

```
'This example gets the location of the ACT! application.
Dim objApp as object
Dim lx as long
Dim ly as long

'Initialize the Application object.
'This starts ACT! if it is not already running.
Set objApp = CreateObject("ACTOLE.APPOBJECT")

'Get the location (Y and Y coordinates of the top left corner)
'of the ACT! application.
objApp.GetPosition lx, ly
List1.AddItem "ACT Position = " & lx & " , " & ly

'Close the Application object
Set objApp = Nothing
```

GetSize Method

Description Gets the size of the application frame window.

Object Application object

Syntax *object*.**GetSize** (*lWidth, lHeight*)

Parameters *lWidth* A long integer (pointer to long integer in Visual C++) representing the width of the main window.

lHeight A long integer (pointer to long integer in Visual C++) representing the height of the main window.

Return type Long Integer

See also GetPosition Method, ReSize Method

ReSize Method in the common application class

Example

```
'This example gets the size of the application frame window.
Dim objApp as object
Dim lHeight as long
Dim lWidth as long

'Initialize the Application object.
'This starts ACT! if it is not already running.
Set objApp = CreateObject("ACTOLE.APPOBJECT")

'Get the dimensions of the window.
objApp.GetSize lWidth, lHeight
List1.AddItem "Width = " & lWidth
List1.AddItem "Height = " & lHeight

'Close the Application object.
Set objApp = Nothing
```

GetUserId Method

Description Gets the Unique ID of the user currently logged on to ACT!

Object Application object

Syntax *object*.**GetUserId**

Return type String/BSTR

GetLastError S_NO_OPENDB, S_NO_SECURITY, S_NO_USER

Example

```
'This example gets the Unique ID of the current user.
Dim objApp as object
Dim sUserId as string

'Initialize the Application object.
'This starts ACT! if it is not already running.
Set objApp = CreateObject("ACTOLE.APPOBJECT")

'Gets the Unique ID of the user currently logged on to ACT!
sUserId = objApp.GetUserId

'Close the Application object.
Set objApp = Nothing
```

GetUserPrivilege Method

Description Gets the security level of the current user of the open database. Returns S_ERROR on failure.

Object Application object

Syntax *object*.**GetUserPrivilege**

Return type Long Integer

Comments The following table lists the values that are returned by this method.

Value	Setting	Value	Setting
0	Browse	2	Administrator
1	Standard		

GetLastError S_NO_SECURITY, S_NO_USER

GetVersion Method

Requires ACT! 4.0 or later

Description Gets the version of the open ACT! database or the ACT! application. To get the version of an ACT! database it must be open. Returns NULL on error.

Object Application object

Syntax *object*.**GetVersion** (*iFor*)

Parameters *iFor* A short integer representing the version to get. Specify 0 to get the version of the open ACT! database or 1 to get the version of the ACT! application.

Return type String/BSTR

Help Method

Description Displays the help window for the help topic specified by the context ID.

Object Application object

Syntax *object*.**Help** [(*szHelpFile, lContextID*)]

Parameters *szHelpFile* A string representing the name of the Help file.

lContextID A long integer representing the context ID of the Help topic to be displayed.

Return type Long Integer

Comments If both szHelpFile and lContextID are specified, the appropriate Help topic displays; otherwise, the default ACT! Help system is used.

IsDBOpen Method

Description Returns True if the database is open or False if it is not open.

Object Application object

Syntax *object*.**IsDBOpen**

Return type Boolean

See also OpenDB Method

Example See GetOpenDBName Method.

IsVisible Method

Description Returns the display state of the application. ReturnsTrue if the main frame window of the application is visible or False if it is not visible.

Object Application object

Syntax *object*.**IsVisible**

Return type Boolean

GetLastError S_MAIN_WND

See also GetSize Method, Maximize Method, Minimize Method, ReSize Method, Show Method

Example See Show.

Maximize Method

Description Maximizes the application frame window. Returns S_ERROR on failure.

Object Application object

Syntax *object*.**Maximize**

Return type Long Integer

GetLastError S_MAIN_WND

See also GetSize Method, Minimize Method, ReSize Method, Show Method

Minimize Method

Description Minimizes the application frame window. Returns S_ERROR on failure.

Object Application object

Syntax *object*.**Minimize**

Return type Long Integer

GetLastError S_MAIN_WND

See also GetSize Method, Maximize Method, ReSize Method, Show Method

OpenDB Method

Description Closes the active database, opens the specified database.

Object Application object

Syntax *object*.**OpenDB** (*szDBName*)

Parameters *szDBName* A string representing the name of the database to be opened.

Return type Object/LPDISPATCH

GetLastError S_INVALID_INPUT

Comments Normally, invoking OpenDB displays the Login dialog box. Invoking this method (or the OpenFile method) before the user responds to the Login dialog box causes unpredictable results.

See also IsDBOpen Method, OpenFile Method

Example

```
'This example opens the ACT5DEMO database.
Dim objApp as object
'Initialize the Application object.
'This starts ACT! if it is not already running.
Set objApp = CreateObject("ACTOLE.APPOBJECT")
objApp.OpenDB C:\My Documents\ACT\Database\ACT5demo.dbf
'Close the Application object.
Set objApp = Nothing
```

OpenFile Method

Description Opens the specified database file. Opening a database file closes the currently active database and makes the new database the active database.

Object Application object

Syntax *object*.**OpenFile** (*szDBName*)

Parameters szDBName A string representing the name of the database file to be opened.

Return type Long Integer

GetLastError S_INVALID_INPUT

Comments Normally, invoking OpenFile displays the Login dialog box. Invoking this method (or the OpenDB method) before the user responds to the Login dialog box causes unpredictable results.

See also IsDBOpen Method, OpenDB Method

Example

```
'This example opens the ACT5DEMO.DBF database file.
Dim objApp as object

'Initialize the Application object.
'This starts ACT! if it is not already running.
Set objApp = CreateObject("ACTOLE.APPOBJECT")
objApp.OpenFile C:\My Documents\ACT\Database\ACT5demo.dbf
objApp.Maximize

'Close the Application object.
Set objApp = Nothing
```

Preferences Method

Description Returns a dispatch pointer to the Preferences object. The returned dispatch pointer can be used to manipulate user preferences within the ACT! application. This method fails if there is no open database. Returns NULL on error.

Object Application object

Syntax *object*.**Preferences**

Return type Object/LPDISPATCH

GetLastError S_NO_OPENDB, S_MEM_ERROR

ProcessFile Method

Description Launches the specified file from the NetLinks folder. Returns S_ERROR on failure.

Object Application object

Syntax *object*.**ProcessFile** (*szFilename, lReserved*)

Parameters *szFilename* A string representing the full path of a file in the NetLinks folder. You must include the full path or the file cannot be located.

lReserved A long integer value reserved for future use.

Return type Long Integer

GetLastError S_ERROR

PurgeHistories Method

Description Purges histories of all contacts and against all activities for the specified database. The database should not be the currently open database. Use this method to maintain the database. Returns S_ERROR on failure.

Object Application object

Syntax *object*.**PurgeHistories** (*szDBName*)

Parameters *szDBName* A string representing the name of the database where histories are to be purged.

Return type Long Integer

GetLastError S_NO_DB, S_INVALID_INPUT

Comments It is possible that some database maintenance operations are occurring in the background after this method returns. Other database maintenance methods should not be invoked immediately after PurgeHistories.

See also PurgeNotes Method, PurgeTransactions Method

Example

```
'This example purges all history records for the CONTACTS database.
Dim objApp as object

'Initialize the Application object.
'This starts ACT! if it is not already running.
Set objApp = CreateObject("ACTOLE.APPOBJECT")

'Open ACT5DEMO.DBF.
objApp.OpenFile C:\My Documents\ACT\Database\ACT5demo.dbf

'Purge all history records for the CONTACTS database.
objApp.PurgeHistories "C:\My Documents\ACT\Database\CONTACTS.dbf"

'Close the Application object.
Set objApp = Nothing
```

PurgeNotes Method

Description Purges notes for all contacts and against all activities for the specified database. The database should not be the currently open database. Use this method to maintain the database. Returns S_ERROR on failure.

Object Application object

Syntax *object*.**PurgeNotes** (*szDBName*)

Parameters *szDBName* A string representing the name of the database where notes are to be purged.

Return type Long Integer

GetLastError S_NO_DB, S_INVALID_INPUT

Comments It is possible that some database maintenance operations are occurring in the background after this method returns. Other database maintenance methods should not be invoked immediately after PurgeNotes.

See also PurgeHistories Method, PurgeTransactions Method

Example

```
'This example purges all notes in the CONTACTS database.
Dim objApp as object
```

```
'Initialize the Application object.
'This starts ACT! if it is not already running.
Set objApp = CreateObject("ACTOLE.APPOBJECT")

'Open ACT5DEMO.DBF.
objApp.OpenFile C:\My Documents\ACT\Database\ACT5demo.dbf

'Purge all notes for the CONTACTS database.
objApp.PurgeNotes C:\My Documents\ACT\Database\CONTACTS.dbf

'Close the Application object.
Set objApp = Nothing
```

PurgeTransactions Method

Description Purges all transactions for all contacts and against all activities for the specified database. The database should not be the currently open database. Use this method to maintain the database. Returns S_ERROR on failure.

Object Application object

Syntax *object*.**PurgeTransactions** (*szDBName*)

Parameters *szDBName* A string representing the name of the database where histories are to be purged.

Return type Long Integer

GetLastError S_NO_DB, S_INVALID_INPUT

Comments It is possible that some database maintenance operations are occurring in the background after this method returns. Other database maintenance methods should not be invoked immediately after PurgeTransactions.

See also PurgeHistories Method, PurgeNotes Method

Example

```
'This example purges all transactions for the CONTACTS database.
Dim objApp as object

'Initialize the Application object.
'This starts ACT! if it is not already running.
Set objApp = CreateObject("ACTOLE.APPOBJECT")

'Open ACT5DEMO.DBF.
objApp.OpenFile C:\My Documents\ACT\Database\ACT5demo.dbf

'Purge all transactions for the CONTACTS database.
objApp.PurgeTransactions C:\My Documents\ACT\Database\ACT5demo.dbf

'Close the Application object.
Set objApp = Nothing
```

ReIndexDB Method

Description Reconstructs the index for the specified database. The database should not be the currently open database. Use this method to maintain the database. Returns S_ERROR on failure.

Object Application object

Syntax *object*.**ReIndexDB** (*szDBName*)

Parameters *szDBName* A string representing the name of the database to be reindexed. The database should not be the current database

Return type Long Integer

GetLastError S_NO_DB, S_INVALID_INPUT

Comments It is possible that some database maintenance operations are occurring in the background after this method returns. Other database maintenance methods should not be invoked immediately after ReIndexDB.

Example See CompressDB Method.

RemoveOutlookActivities Method

Requires ACT! 5.0 or later

Description Removes Outlook activities from ACT!, or ACT! Activities from Outlook, or both depending on the parameters used. This method applies only when Outlook is installed. Returns 0 if successful or returns an error code if unsuccessful. For more information, see Error codes.

Object Application object

Syntax *object*.**RemoveOutlookActivities** (*iRemoveIn*)

Parameters *iRemoveIn* A short integer representing the activities to remove.

The following table lists the value for each activity type:

Value	Activities
1	Removes Outlook activities from ACT!
2	Removes ACT! activities from Outlook
3	Removes Outlook activities from ACT! and removes ACT! activities from Outlook

Return type Long Integer

See also UpdateOutlookActivities Method

ReSize Method

Description Sets the size of the application main frame window. The current (top-left) position of the frame is unchanged. Returns S_ERROR on failure.

Object Application object

Syntax *object*.**ReSize** (*iHeight, iWidth)*

Parameters *iHeight* A short integer representing the new height of the frame.

iWidth A short integer representing the new width of the frame.

Return type Long Integer

GetLastError S_MAIN_WND

See also GetPosition Method, GetSize Method, Maximize Method, Minimize Method, Show Method

Example

```
'This example resizes the application window.
Dim objApp as object

'Initialize the Application object.
'This starts ACT! if it is not already running.
Set objApp = CreateObject("ACTOLE.APPOBJECT")
'Resize the application window.
objApp.ReSize 500, 800
'Close the Application object.
Set objApp = Nothing
```

RestoreDB Method

Requires ACT! 4.0 or later

Description Restores the database from the specified zip file name and location. Returns S_ERROR on failure.

Object Application object

Syntax *object*.**BackupDB** (*szSourceZipFile, szDestinationPath, iMode*)

Parameters *szSourceZipFile*A string representing the source name and path of the zip file containing the backed up database to be restored.

*szDestinationPath*A string representing the destination name and path for the database to be restored.

iMode Specify 167 to overwrite any existing database located as specified in the szDestinationPath parameter. Specify 168 to not overwrite an existing database and return without restoring the backup if an existing database is found in the specified location.

Return type Long Integer

See also BackupDB Method

RunMacro Method

Description Runs the macro stored in the specified file at recorded speed. Returns S_ERROR on failure.

Object Application object

Syntax *object*.**RunMacro** (*szMacroName*)

Parameters *szMacroName*A string representing the full path and file name of the file containing the macro.

Return type Long Integer

GetLastError S_INVALID_INPUT, S_MACRO_ERROR

Example

```
'This example runs the MYMACRO macro.
Dim objApp as object

'Initialize the Application object.
'This starts ACT! if it is not already running.
Set objApp = CreateObject("ACTOLE.APPOBJECT")

'Open the database.
objApp.OpenFile C:\My Documents\ACT\Database\ACT5demo.dbf

'Run the macro MYMACRO. (Macro files have extension .MPR and
'are stored in C:\PROGRAM FILES\ACT\MACRO by default.)
objApp.RunMacro "C:\PROGRAM FILES\ACT\MACRO\MYMACRO.MPR"

'Close the Application object.
Set objApp = Nothing
```

SaveCurrentLookup Method

Description Saves the current lookup in the specified file. If a file with the same name exists, it is overwritten. Returns S_ERROR on failure.

Object Application object

Syntax *object*.**SaveCurrentLookup** (*szFilename*)

Parameters *szFilename* A string representing the name of the file where the lookup should be saved.

Return type Long Integer

GetLastError S_CON_DOC, S_CON_LIST, S_ERROR, S_FILE_OPEN

SendKey Method

Description Sends a message to the application. Similar to Windows SendMessage. Currently only WM_CHAR messages are sent.

Object Application object

Syntax *object*.**SendKey** (*lParam1, lParam2, lParam3*)

Parameters *lParam1* A long integer specifying additional data to be sent.

lParam2 A long integer specifying additional data to be sent. (Currently ignored.)

lParam3 A long integer specifying additional data to be sent. (Currently ignored.)

Return type Long Integer

Comments The returned value depends on the message being sent.

Show Method

Description Hides or displays the application frame window. Returns S_ERROR on failure.

Object Application object

Syntax *object*.**Show** (*True*|*False*)

Parameters *True*|*False* Specify True to display the frame or False to hide it.

Return type Long Integer

GetLastError S_ERROR, S_MAIL_WND

Comments After the application frame window is hidden, any action that causes a dialog box to be displayed might cause the frame window to become visible again.

See also GetSize Method, Maximize Method, Minimize Method, ReSize Method

Example

```
'This example displays the application window.
Dim objApp as object

'Initialize the Application object.
'This starts ACT! if it is not already running.
Set objApp = CreateObject("ACTOLE.APPOBJECT")

'Open the database.
objApp.OpenFile C:\My Documents\ACT\Database\ACT5demo.dbf

'If the application is not visible, then display it.
If objApp.IsVisible = False then
   objApp.Show (True)
End If

'Close the Application object.
Set objApp = Nothing
```

Update Method

Description Redraws the application frame window. Returns S_ERROR on failure.

Object Application object

Syntax *object*.**Update**

Return type Long Integer

UpdateOutlookActivities Method

Requires ACT! 5.0 or later

Description Updates ACT! with Outlook activities, or Outlook with ACT! activities, or both depending on the parameters used. This method applies only when Outlook is installed. Returns 0 if successful or returns an error code if unsuccessful. For more information, see Error codes.

Object Application object

Syntax *object*.**UpdateOutlookActivities** (*iUpdateDirection, iDurationType*)

Parameters *iUpdateDirection*A short integer representing the direction of the update.

The following table lists the values for this parameter:

Value	Direction
1	From Outlook! to ACT!
2	From ACT! to Outlook
3	Both

iDurationType A short integer representing the duration type of the update.

The following table lists the values for this parameter:

Value	Duration
1	All days
2	Today
3	Today and in the future
4	Over the next 30 days

Return type Long Integer

See also RemoveOutlookActivities Method

Example

```
'The following line updates ACT! with activities in Outlook for today.
ret = objApp.UpdateOutlookActivities(1, 2)
```

Views Method

Description Returns a dispatch pointer to the view collection object. The returned dispatch pointer can be used to create new views and access and manipulate any existing views in the application. This method fails if there is no open database. Returns NULL on error.

Object Application object

Syntax *object*.**Views**

Return type Object/LPDISPATCH

GetLastError S_NO_OPENDB, S_MEM_ERROR

Example

```
'This example creates a Views object.

Dim objApp as object
Dim objViews as object

'Initialize the Application object.
'This starts ACT! if it is not already running.
Set objApp = CreateObject("ACTOLE.APPOBJECT")

'Open the database.
objApp.OpenFile C:\My Documents\ACT\Database\ACT5demo.dbf

'Create a Views object.
Set objViews = App.Views

'Close the Application object.
Set objApp = Nothing
```

CalendarView object

The CalendarView object provides an interface to use the Calendar view in the ACT! application. The following properties and methods apply only to the CalendarView object. See Common properties and methods for properties and additional methods that apply to this object.

Sample Code

The following sample code demonstrates how to use the CalendarView object:

```
Dim objApp as object
Dim objViews as object
Dim objCal as object
Dim i as long

'Initialize the Application object.
'This starts ACT! if it is not already running.
Set objApp = CreateObject("ACTOLE.APPOBJECT")

'Open the database.
objApp.OpenFile C:\My Documents\ACT\Database\ACT5demo.dbf

'Create a Views object.
Set objViews = App.Views

'Create the Calendar object.
Set objCal = objViews.Create(5, "MyCal")

'If the Calendar Active month is not April, then set it to April.
If objCal.GetActiveMonth <> 4 Then
   objCal.SetActiveMonth (4)
End If

'If the Calendar is not weekly, then set it to weekly.
If objCal.GetCalendarMode <> 301 Then
   objCal.SetCalendarMode (301)
End If
```

```
objViews.CloseAll
Set objCal = Nothing
Set objViews = Nothing

'Close the Application object.
Set objApp = Nothing
```

Methods

Name	Parameter(s)	Parameter type(s)	Return type
GetActiveMonth Method	None	*	Short Integer
GetCalendarMode Method	None	*	Short Integer
SetActiveMonth Method	iMonth	Short Integer	Long Integer
SetCalendarMode Method	iMode	Short Integer	Long Integer

GetActiveMonth Method

Description Returns the month between 1 and 12 indicating the active in the calendar view. Returns 0 (zero) on error.

Object CalendarView object

Syntax *object*.**GetActiveMonth**

Return type Short Integer

GetLastError S_NO_CALVW

See also SetActiveMonth Method

GetCalendarMode Method

Description Returns the mode in which the calendar is being displayed.

Object CalendarView object

Syntax *object*.**GetCalendarMode**

Return type Short Integer

Comments The following table lists the values that are returned by this method.

Value	Mode	Value	Mode
300	Monthly calendar	302	Daily calendar
301	Weekly calendar	0	Error

GetLastError S_NO_CALVW

See also SetCalendarMode Method

SetActiveMonth Method

Description Sets the specified month to be the active month in the calendar. Returns S_ERROR on an invalid month.

Object CalendarView object

Syntax *object*.**SetActiveMonth** (*iMonth*)

Parameters *iMonth* A short integer between 1 and 12 representing the active month.

Return type Long Integer

GetLastError S_NO_CALVW, S_INVALID_INPUT

See also GetActiveMonth Method

SetCalendarMode Method

Description Sets the mode in which the calendar is displayed in the calendar view and activates the view.

Object CalendarView object

Syntax *object*.**SetCalendarMode** (*iMode*)

Parameters *iMode* A short integer representing the mode for displaying the calendar:

Value	Mode	Value	Mode
300	Monthly	302	Daily
301	Weekly		

If iMode is not one of the above values, the mode is set to Monthly.

Return type Long Integer

GetLastError S_NO_CALVW

See also GetCalendarMode

ContactListView object

The ContactListView object provides an interface to use the Contact List view in the ACT! application. The following methods apply only to the ContactListView object. See Common properties and methods for properties and additional methods that apply to this object.

Methods

Name	Parameter(s)	Parameter type(s)	Return type
AddNewContactEx Method	None	*	String/BSTR
GetGrid Method	None	*	Object/LPDISPATCH

AddNewContactEx Method

Requires ACT! 4.0 or later

Description Adds a new contact to the database. Use SetField to set the fields in the newly created row. Returns the Unique ID of the new contact. Use GetRowNumber to find the contact's row number.

Object ContactListView object

Syntax *object*.**AddNewContactEx**

Return type String/BSTR

Comments The new contact appears as the first visible row in the Contact List view. The Contact List view is activated after a call to this method.

See also GetField Method, GetRowNumber Method, SetField Method in the Grid object

Example

```
'This example adds a new contact in the Contact List view.
Dim objCtListView As Object
Dim objCtView As Object
Dim objCtListGrid As Object
Dim RN As Integer
Dim objV As Object
Dim uid As String
Dim objApp As Object
```

```
'Initialize the Application object.
'This starts ACT! if it is not already running.
Set objApp = CreateObject("ACTOLE.APPOBJECT")

'Open the database.
objApp.OpenDB C:\My Documents\ACT\Database\ACT5demo.dbf

'Create the Views object.
Set objViews = objApp.Views
'Create the Contact List view.
Set objCtListView = objViews.Create(2, "CL")
'Create the Grid object.
Set objCtListGrid = objCtListView.GetGrid

'Add a new Contact record and get the Unique ID.
uid = objCtListView.AddNewContactEx
'Get the row number.
RN = objCtListGrid.GetRowNumber(uid)

'Populate the fields.
objCtListGrid.SetField CF_Name, RN, "Carl Bowman"
RN = objCtListGrid.GetRowNumber(uid)
objCtListGrid.SetField CF_Company, RN, "Cordoba Coffee Shops"
RN = objCtListGrid.GetRowNumber(uid)
objCtListGrid.SetField CF_Phone, RN, "415-555-1212"
RN = objCtListGrid.GetRowNumber(uid)
objCtListGrid.SetField CF_Title, RN, "Engineer"
Set objCtListGrid = Nothing

'Close the Contact List view.
objCtListView.Close
Set objCtListView = Nothing
Set objViews = Nothing
'Close the Application object.
Set objApp = Nothing
```

GetGrid Method

Description Returns a dispatch pointer to a grid object for the Contact List view. The returned dispatch pointer can be used to manipulate the grid object. Returns NULL on error.

Object ContactListView object

Syntax *object*.**GetGrid**

Return type Object/LPDISPATCH

GetLastError S_ERROR

Example

```
'This example returns a dispatch pointer.
Dim objApp as object
Dim objViews as object
Dim objContactListView as object
Dim objContactListViewGrid as object

'Initialize the Application object.
'This starts ACT! if it is not already running.
Set objApp = CreateObject("ACTOLE.APPOBJECT")
```

```
'Open the database.
objApp.OpenFile C:\My Documents\ACT\Database\ACT5demo.dbf

'Initialize the Views object.
Set objViews = objApp.Views

List1.AddItem objApp.GetLastError
Set objCtListView = objViews.Create(2, "CL")

'Get the Grid Object in the Contact List.
Set objCtListGrid = objCtListView.GetGrid

'List the contact name and company for all the contacts in the Contact List.
For i = 0 To objCtListGrid.GetRowCount - 1
    List1.AddItem objCtListGrid.GetField(CF_Name, i) & "  ,  " &
        objCtListGrid.GetField(CF_Company, i)
Next i

Set objCtListGrid = Nothing
objCtListView.Close
Set objCtListView = Nothing
Set objViews = Nothing
Set objApp = Nothing
```

ContactView object

The ContactView object provides an interface to use the Contact view in the ACT! application. The following methods apply only to the ContactView object. See Common properties and methods for properties and additional methods that apply to this object.

Methods

Name	Parameter(s)	Parameter type(s)	Return type
Activities Method	None	*	Object/LPDISPATCH
AddContactToGroup Method	szGroupID	String	Long Integer
AddNewActivityEx Method	szContactID…,szDateTime,iType,iTimeless,szRegarding,lDuration	String,String,Short Integer,Short Integer,String,Long Integer	String/BSTR
AddNewContact Method	None	*	Long Integer
AddNoteHistoryEx Method	iType	Short Integer	String/BSTR
AttachFile Method	szFilename	String	Long Integer
BOL Method	None	*	Boolean
CompleteSale Method	szSaleID,iStatus,dateClosed,szReason,lUnits,fUnitPrice,fAmount	StringShort IntegerDateStringLong IntegerFloat (Single)Float (Single)	Long Integer
CreateLookup Method	szContact…	String	Long Integer

Name	Parameter(s)	Parameter type(s)	Return type
CreateSalesForecast Method	szContactID,szProductID,szType,szCompetitor,lUnits,fUnitPrice,fAmount,dateClose,lProb	String,String,String,String ,Long Integer,Float (Single),Float (Single),Date,Long Integer	String
Delete Method	None	*	Long Integer
DeleteContactFast Method	None	*	Long Integer
EOL Method	None	*	Boolean
GetActiveGroup Method	None	*	String/BSTR
GetActiveGroupName Method	None	*	String/BSTR
GetActiveTab Method	None	*	String/BSTR
GetCount Method	None	*	Short Integer
GetCurrentID Method	None	*	String/BSTR
GetField Method	lFieldID	Long Integer	String/BSTR
GetTabCount Method	None	*	Short Integer
GetTabName Method	iTabNo	Short Integer	String/BSTR
Goto Method	szContactID	String	Long Integer
GroupMembership Method	None	*	Object/LPDISPATCH
LookupAll Method	None	*	Long Integer
LookupFieldEx Method	lFieldID szFieldValue iType	Long Integer String Short Integer	Long Integer
LookupMyRecord Method	None	*	Long Integer
LookupPrevious Method	None	*	Long Integer
MoveFirst Method	None	*	Long Integer
MoveLast Method	None	*	Long Integer
MoveNext Method	None	*	Long Integer
MovePrevious Method	None	*	Long Integer
NewContactDialog Method	None	*	Long Integer
NotesHistory Method	None	*	Object/LPDISPATCH
RunQuery Method	szQueryFile	String	Long Integer
Sales Method	None	*	Object/LPDISPATCH
SaveQuery Method	szQueryFile	String	Long Integer
SelectContactDlg Method	szText	String	String/BSTR
SetActiveGroup Method	szGroupID	String	Long Integer
SetActiveGroupName Method	szGroupName	String	Long Integer
SetActiveTab Method	szTabName	String	Long Integer
SetField Method	lFieldID szFieldValue	Long Integer String	Long Integer
TriggerActivitySeries Method	iLookupType SeriesDate szActivitySeriesName	Short Integer Date String	Long Integer

Activities Method

Description Returns a dispatch pointer to a Grid object for the Activities tab. The Activities tab must be the active tab before calling this method. Returns NULL on error.

Object ContactView object

Syntax *object.***Activities**

Return type Object/LPDISPATCH

GetLastError S_MEM_ERROR, S_NO_DALLIST_VIEW, S_UNKNOWN

See also GetActiveTab Method, NotesHistory Method, SetActiveTab Method

AddContactToGroup Method

Description Adds the current contact to the specified group. If the function is successful, the specified group appears in the Groups tab of the Contact view. If the contact is currently a member of the specified group or the Unique ID is invalid, this method fails. Returns S_ERROR on failure.

Object ContactView object

Syntax *object.***AddContactToGroup** (*szGroupID*)

Parameters *szGroupID* A string representing the Unique ID of the group.

Return type Long Integer

GetLastError S_ADD_CONTACT, S_DUPLICATE_GRP, S_INVALID_INPUT, S_UNKNOWN

See also AddMemberToGroup Method in GroupView object

Example

```
'This example adds the current contact to the group with the ID GroupID.
Dim objApp as object
Dim objViews as object
Dim objContact as object

'Initialize the Application object.
'This starts ACT! if not already running.
Set objApp = CreateObject("ACTOLE.APPOBJECT")

'Open the database.
objApp.OpenFile C:\My Documents\ACT\Database\ACT5demo.dbf

'Create a Views object.
Set objViews = objApp.Views
'Create the ContactView object.
Set objContact = objViews.Create(1, "MyContact")

'Set Groups to be the active tab.
objContact.SetActiveTab "Groups"

'Add the current contact in the Contact view to the group
'with the Unique ID GroupID.
objCView.AddContactToGroup GroupId

'Close the Contact view.
objContact.Close
Set objContact = Nothing
Set objViews = Nothing
'Close the Application object.
Set objApp = Nothing
```

AddNewActivityEx Method

Requires ACT! 4.0 or later

Description Adds a new activity for the specified contacts using Activity defaults set in Preferences. Returns the Unique ID of the new activity on success and NULL on failure. If any ID in szContactID is invalid, this method returns S_OK, but GetLastError returns S_INVALID_INPUT. Use GetFilter Method and SetFilter Method in the Grid object before using this method.

Object ContactView object

Syntax *object*.**AddNewActivityEx** (*szContactID…, szDateTime, iType, iTimeless, szRegarding, lDuration*)

Parameters *szContactID* A string representing the Unique ID(s) of contacts with which the new activity will be associated. If a NULL value is specified for this required parameter, the new activity will be associated with the default of the current database user.

Note: Specify multiple Unique IDs as a continuous string of 12?character values, with no delimiters between the Unique ID values.

szDateTime A string representing the starting date and time of the activity, formatted in Windows Regional Settings Short Date style and Time style.

iType A short integer representing the type of activity to be added.

The following table lists the values for this parameter.

Value	Setting	Value	Setting
0	Call	2	To-do
1	Meeting		

iTimeless A short integer representing the timeless status of the activity to be added.

The following table lists the values for this parameter.

Value	Setting
0	Not timeless
1	Timeless

szRegarding A string representing the description of the activity.

lDuration A long integer representing the duration in minutes of the activity.

Return type String/BSTR

GetLastError S_ACTIVITY_INIT_FAIL, S_CON_DOC, S_HELPER_EEDIT_FAIL, S_INVALID_INPUT, S_PROPS_NOT_FOUND, S_UNKNOWN

Example

```
'This example adds a high priority to-do to the current contact, dated
 '10/20/98, at 10:00 AM, duration 45 minutes, regarding "A Meeting".

'Initialize the application object.
Set objApp = CreateObject("ACTOLE.APPOBJECT")

'Create the Views object.
Set objViews = objApp.Views

'Create the Contacts view.
Set objCView = objViews.Create(1, "CV")
```

```
'Get the Unique ID of the current contact.
sContact = objCView.GetCurrentID
'Set Activities to be the Active Tab
objCView.SetActiveTab "Activities"

'Create the Grid object.
Set objActiv = objCView.Activities
'Get Activities filter values.
objActiv.GetFilter x, y, z
j = 1 or 2 or 4 or 8 or 16 or 32 or 64 or 128
objActiv.SetFilter j, 1, " "

'Add an activity. Its Unique ID is returned in uid as a string.
uid = objCView.AddNewActivityEx(sContact, "10/20/98 10:00AM",
    activitytype_meeting,0,"A meeting", "45")
'Get the row number.
RN = objActiv.GetRowNumber(uid)

'Set other parameters in the activity.
objActiv.SetField AF_Priority, RN, activitypriority_medium

'Reset the Activities filter values.
objActiv.SetFilter x, y, z

Set objActiv = Nothing
Set objCView = Nothing
```

AddNewContact Method

Description Displays a new contact with all fields blank. Use SetField to populate the contact fields.

Object ContactView object

Syntax *object.***AddNewContact**

Return type Long Integer

GetLastError S_UNKNOWN

See also Delete Method, GetField Method, SetField Method

Example

```
'This example displays the Contact view and adds a contact.
Dim objApp as object
Dim objViews as object
Dim objContact as object

'Initialize the Application object.
'This starts ACT! if it is not already running.
Set objApp = CreateObject("ACTOLE.APPOBJECT")

'Open the database.
objApp.OpenFile C:\My Documents\ACT\Database\ACT5demo.dbf

'Create a Views object.
Set objViews = App.Views

'Create the ContactView object -This brings up the Contact view.
Set objContact = objViews.Create(1, "MyContact")
```

```
'Add a contact with all blank fields. Set the Contact, Company,
'address and Phone fields.
objContact.AddNewContact
objCView.SetField CF_Name, "My Contact"
objCView.SetField CF_Company, "My Company"
objCView.SetField CF_Address1, "My Address1"
objCView.SetField CF_City, "City"
objCView.SetField CF_State, "State"
objCView.SetField CF_Phone, "415-555-1212"
objCView.SetField CVF_EmailAddress, "MyContact@MyCompany.com"

'Close the Contact view.
objContact.Close
objViews.CloseAll
Set objContact = Nothing
Set objViews = Nothing
'Close the Application object.
Set objApp = Nothing
```

AddNoteHistoryEx Method

Requires ACT! 4.0 or later

Description Adds a Notes/History record for the current contact. Use SetField in the Grid object to set other fields in the newly created row. The Notes/History tab must be the active tab before calling this method. Returns the Unique ID of the new Notes/History record on success and NULL on failure. Use GetFilter and SetFilter in the Grid object before using this method.

Note: This method replaces the functionality of the AddNoteHistory method.

Object ContactView object

Syntax *object*.**AddNoteHistoryEx** (*iType*)

Parameters *iType* A short integer indicating whether a note or the type of history to be added. To add a note, iType must be 100.

To add a history, iType must be one of the following values:

Value	Setting	Value	Setting
0	Call attempted	7	Meeting not held
1	Call completed	8	To-do done
2	Call received	9	To-do not done
6	Meeting held	17	Left message

Return type String/BSTR

GetLastError S_CON_DOC, S_INVALID_INPUT, S_MEM_ERROR, S_NO_DALLIST_VIEW, S_NO_RECORD, S_RECORD_HISTORY, S_UNKNOWN

See also GetField Method, SetField Method in Grid object

AddNoteEx Method in GroupView object

Example

```
'This example adds a note to the current contact.
'Initialize the application object.
Set objApp = CreateObject("ACTOLE.APPOBJECT")

'Create the Views object.
Set objViews = objApp.Views
```

```
'Create the Contact view.
Set objCView = objViews.Create(1, "CV")

'Get the Unique ID of the Current contact.
sContact = objCView.GetCurrentID

'Set Notes/History to be the active tab.
objCView.SetActiveTab "Notes/History"

Create the Grid object.
Set objNH = objCView.NotesHistory

'Add a note. The Unique ID is returned in uid as a string.
uid = objCView.AddNoteHistoryEx(100)

'Get the row number.
RN = objNH.GetRowNumber(uid)
objNH.SetField NHF_Text, RN, " Test Note"
objNH.SetField NHF_UserTime, RN, "10/28/98 8:00AM"

Set objNH = Nothing
Set objCView = Nothing
```

AttachFile Method

Requires ACT! 4.0 or later

Description Adds the specified file as an attachment to the Notes/History tab of the Contacts view. Returns the ACT! unique ID assigned to the attached file when successful, or an empty string on failure.

Object ContactView object

Syntax *object*.**AttachFile** (*szFilename*)

Parameters *szFilename* A string representing the complete path and the file name of the file to add as an attachment.

Return type String

GetLastError S_CON_DOC, S_ERROR, S_NOT_FOUND, S_UNKNOWN

See also AttachFile Method in GroupView object

BOL Method

Description Returns True if the current contact is at the beginning of the lookup or False if it is not at the beginning of the lookup or if an error occurs.

Object ContactView object

Syntax *object*.**BOL**

Return type Boolean

GetLastError S_CON_DOC, S_UNKNOWN

See also EOL Method, Goto Method, MoveFirst Method, MoveLast Method, MoveNext Method, MovePrevious Method

CompleteSale Method

Requires ACT! 5.0 or later

Description Completes the specified sale. Returns 0 if successful or an error code if unsuccessful. For more information, see Error codes.

Object ContactView object

Syntax *object*.**CompleteSale** (*szSaleID, iStatus, dateClosed, szReason, lUnits, fUnitPrice, fAmount*)

Parameters *szSaleID* A string representing the Unique ID of the sales opportunity record for which to complete the sale.

iStatus A short integer representing the status of the sale.

The following table lists the values for this parameter:

Value	Status	Value	Status
0	Open	2	Lost
1	Won		

dateClosed The date the sale closed, formatted in Windows Regional Settings Short Date style.

szReason A string representing the reason the sale was won or lost.

lUnits A long integer representing the number of units sold.

fUnitPrice A float (single) representing the price at which each unit was sold.

fAmount A float (single) representing the total amount of the sale.

Return type Long Integer

See also CreateSalesForecast, Sales

Example

```
'The following example code closes a particular sale.
Set objViews = objApp.Views
Set objContactView = objViews.Create(1, "CL")
objContactView.Goto ContactId
objContactView.SetActiveTab "Sales/Opportunities"
Set objSales = objContactView.Sales
    'Get the Sale Unique ID for a particular row number.
    SaleId = objSales.GetUniqueID(RowNumber)

'If the sales is not closed, then complete it.
If objSales.GetField(SLF_Status, i) = "Sales Opportunity" Then
'Won the Sale, new close date 8/6/99, new unit price 10.85,
'quantity 6000.
    objContactView.CompleteSale SaleId, 1, "8/06/99",
        "We lowered the price to beat the competition", 6000, 10.85,
        (10.85 * 6000)
End If
```

CreateLookup Method

Requires ACT! 5.0 or later

Description Creates a lookup of the list of Contact Unique IDs concatenated without delimiters. Returns 0 if successful or returns an error code if unsuccessful. For more information, see Error codes.

Object ContactView object

Syntax *object*.**CreateLookup** (*szContactID…*)

Parameters *szContactID* A string representing the Unique IDs of the contacts.

Note: Specify multiple Unique IDs as a continuous string of 12-character values, with no delimiters between the Unique ID values.

Return type Long Integer

Example

```
'The following sample creates a lookup of 3 contacts whose Unique IDs
'you have.

Set objViews = objApp.Views
Set objContactView = objViews.Create(1, 2)

'Moving first.
objContactView.MoveFirst
'Concatenate the Unique IDs of the contacts to include in the lookup.
ContactId = ContactId1 & ContactId2 & ContactId3
'Create the lookup of the 3 contacts.
objContactView.CreateLookup ContactId
```

CreateSalesForecast Method

Requires ACT! 5.0 or later

Description Creates a new sales opportunity and returns its Unique ID.

Object ContactView object

Syntax *object*.**CreateSalesForecast** (*szContactID, szProductID, szType, szCompetitor, lUnits, fUnitPrice, fAmount, dateClose, lProb*)

Parameters

szContactID A string representing the Unique ID of the contact to which to associate the sales opportunity.

szProductID A string representing the Unique ID of the product.

szType A string representing the Unique ID of the product type to which to associate the sales record.

szCompetitor A string representing the name of the main competitor for the sale.

lUnits A long integer representing the number of units expected to be sold.

fUnitPrice A float (single) representing the price at which each unit is expected to be sold.

fAmount A float (single) representing the expected total amount of the sale.

dateClose The date the sale is expected to close.

lProb A long integer representing the probability of winning the sale in a percentage value between 0 and 100.

Return type String

See also CompleteSale Method, Sales Method

Example

```
'The following code creates a new Sales Opportunity for 6000 pcs of
'ProductA of "TypeA"@$11 each and with main competitor BigCorp, Inc.
Set objViews = objApp.Views
Set objContactView = objViews.Create(1, "CL")
objContactView.Goto ContactId
objContactView.SetActiveTab "Sales/Opportunities"
Set objSales = objContactView.Sales
'New Sales Opportunity for ProductA of TypeA , 6000 units at $11 each,
'close date 10/22/99 and 65% probability.
sid = objContactView.CreateSalesForecast(ContactId, "ProductA",
    "TypeA", "BigCorp, Inc.", 6000, 11#, (11# * 1000), "10/22/99", 65)
'Get the row number.
RN = objSales.GetRowNumber(sid)
```

```
'Set the "SalesStage"
objSales.SetField SLF_SalesStage, RN, "New Opportunity"
RN = objSales.GetRowNumber(sid)
'Set the start date for the sale.
objSales.SetField SLF_SaleStartDate, RN, "4/4/99"
```

Delete Method

Description Deletes the selected contact and all details about the contact including activities, notes and histories. Displays a dialog box to confirm the deletion.

Object ContactView object

Syntax *object*.**Delete**

Return type Long Integer

GetLastError S_UNKNOWN

See also AddNewContact Method

DeleteContactFast Method

Requires ACT! 5.0 or later

Description Deletes the current contact without invoking the confirmation dialog boxes. Returns 0 if the contact is successfully deleted or an error code if unsuccessful. For more information, see Error codes.

Object ContactView object

Syntax *object*.**DeleteContactFast**

Return type Long Integer

See also DeleteGroupFast Method in GroupView object

Example

```
'The following sample selects a contact and then deletes it.
Set objContactView = objViews.Create(1, "CL")
objContactView.MoveFirst
'Select the contact you want to delete.
ContactId = objContactView.SelectContactDlg("Select Contact you would like
   to delete")
objContactView.Goto ContactId
objContactView.GetField (CF_Name)
'Create a confirmation message, since ACT! will now not provide one.
ret = MsgBox("Are you sure you want to delete " & ContactName & " ?", vbYesNo
 Or vbExclamation)
'If the user selected Yes, delete the contact.
If ret = vbYes Then
   objContactView.DeleteContactFast
End If
```

EOL Method

Description Returns True if the current contact is at the end of the lookup or False if it is not at the end of the lookup or if an error occurs.

Object ContactView object

Syntax *object*.**EOL**

Return type Boolean

GetLastError S_CON_DOC, S_UNKNOWN

See also BOL Method, Goto Method, MoveFirst Method, MoveLast Method, MoveNext Method, MovePrevious Method

GetActiveGroup Method

Description Gets the Unique ID of the active group. If the active group is <No Group>, an empty string or blank is returned.

Note: If you are using a version of ACT! prior to ACT! 5.0.2, this method activates the Contact View upon return. For ACT! 5.0.2 or later, this method does not activate the Contact View. Use the Activate method to activate the Contact View if necessary.

Object ContactView object

Syntax *object*.**GetActiveGroup**

Return type String/BSTR

GetLastError S_CON_DOC, S_UNKNOWN

See also GetActiveGroupName Method, SetActiveGroup Method, SetActiveGroupName Method

Example

```
Dim objApp as object
Dim objViews as object
Dim objContact as object
Dim sGUID as string

'Initialize the Application object.
'This starts ACT! if it is not already running.
Set objApp = CreateObject("ACTOLE.APPOBJECT")

'Open the database.
objApp.OpenFile C:\My Documents\ACT\Database\ACT5demo.dbf

'Create a Views object.
Set objViews = objApp.Views

'Create the ContactView object.
Set objContact = objViews.Create(1, "MyContact")

'If the active group is not Activities, make it the active group.
If objContact.GetActiveGroup <> sGUID Then
 objContact.SetActiveGroup sGUID
End If

'Close the Contact view.
objContact.Close
Set objContact = Nothing
Set objViews = Nothing
'Close the Application object.
Set objApp = Nothing
```

GetActiveGroupName Method

Description Gets the name of the active group. One group is designated as the active group in ACT! All new contacts are associated with the active group by default. Calling this method before adding a contact provides information about the group that the new contact will be associated with. Returns NULL on error.

Object ContactView object

Syntax *object*.**GetActiveGroupName**

Return type String/BSTR

GetLastError S_CON_DOC, S_ERROR, S_UNKNOWN

See also GetActiveGroup Method, SetActiveGroup Method, SetActiveGroupName Method

Example

```
Dim objApp as object
Dim objViews as object
Dim objContact as object
Dim sGUID as string

'Initialize the Application object.
'This starts ACT! if it is not already running.
Set objApp = CreateObject("ACTOLE.APPOBJECT")

'Open the database.
objApp.OpenFile C:\My Documents\ACT\Database\ACT5demo.dbf

'Create a Views object.
Set objViews = objApp.Views

'Create the ContactView object.
Set objContact = objViews.Create(1, "MyContact")

'If the active group is not "Nuts and Things"
'then set the active group to "Nuts and Things".
If objContact.GetActiveGroupName <> "Nuts and Things" Then
   objContact.SetActiveGroupName ("Nuts and Things")
End If

'Close the Contact view.
objContact.Close
Set objContact = Nothing
Set objViews = Nothing
'Close the Application object.
Set objApp = Nothing
```

GetActiveTab Method

Description Gets the text that appears on the currently active tab. Returns NULL on error.

Object ContactView object

Syntax *object*.**GetActiveTab**

Return type String/BSTR

GetLastError S_CON_DOC, S_NO_TAB_CONTAINER, S_UNKNOWN

See also Activities Method, GroupMembership Method, NotesHistory Method, SetActiveTab Method

Example

```
'This example makes the Activities tab active.
Dim objApp as object
Dim objViews as object
Dim objContact as object

'Initialize the Application object.
'This starts ACT! if it is not already running.
Set objApp = CreateObject("ACTOLE.APPOBJECT")

'Open the database.
objApp.OpenFile C:\My Documents\ACT\Database\ACT5demo.dbf
```

```
'Create a Views object.
Set objViews = objApp.Views

'Create the ContactView object.
Set objContact = objViews.Create(1, "MyContact")

'If the active tab is not Activities, set the active tab to Activities.
If objContact.GetActiveTab <> "Activities" Then
   objContact.SetActiveTab "Activities"
End If

'Close the Contact view.
objContact.Close
Set objContact = Nothing
Set objViews = Nothing
'Close the Application object.
Set objApp = Nothing
```

GetCount Method

Description Gets the total number of contacts in the current lookup. Returns -1 on error. Returns all contacts if no records are found.

Object ContactView object

Syntax *object*.**GetCount**

Return type Short Integer

GetLastError S_COUNT, S_UNKNOWN

GetCurrentID Method

Description Returns the Unique ID of the current contact in the lookup. Returns NULL on error.

Note: If you are using a version of ACT! prior to ACT! 5.0.2, this method activates the Contact View upon return. For ACT! 5.0.2 or later, this method does not activate the Contact View. Use the Activate method to activate the Contact View if necessary.

Object ContactView object

Syntax *object*.**GetCurrentID**

Return type String/BSTR

GetLastError S_ERROR, S_UNKNOWN

Example

```
'This example lists all contacts and their Unique IDs.
Dim objApp as object
Dim objViews as object
Dim objContact as object
Dim i as integer

'Initialize the Application object.
'This starts ACT! if it is not already running.
Set objApp = CreateObject("ACTOLE.APPOBJECT")

'Open the database.
objApp.OpenFile C:\My Documents\ACT\Database\ACT5demo.dbf

'Create a Views object.
Set objViews = App.Views
```

```
'Create the ContactView object. This brings up the Contact view.
Set objContact = objViews.Create(1, "MyContact")
objContact.MoveFirst'Move to the first contact.

'Go through all the contacts and list their Unique IDs.
For i = 1 To objContact.GetCount'Get the number of contacts.
    List1.AddItem objContact.GetCurrentID'List contact Unique ID.
    List1.AddItem objContact.GetField (CF_Name)
    objContact.MoveNext
Next i

'Close the Contact view.
objContact.Close
Set objContact = Nothing
Set objViews = Nothing
'Close the Application object.
Set objApp = Nothing
```

GetField Method

Description Gets the value of the specified field in the current record. Returns NULL on failure.

Object ContactView object

Syntax *object*.**GetField** (*lFieldID*)

Parameters *lFieldID* A long integer representing the name of the field whose value is to be retrieved. See See ACT! Databases for lists of field IDs and names (Field constants).

Return type String/BSTR

GetLastError S_CON_DOC, S_UNKNOWN

See also SetField Method

Example See GetCurrentID Method.

GetTabCount Method

Requires ACT! 4.0 or later

Description Returns the number of tabs in the Contact view, including tabs added using the Adding Extensible Views and Tabs to ACT! component of the ACT! SDK. Returns -1 on error.

Object ContactView object

Syntax *object*.**GetTabCount**

Return type Short Integer

GetLastError S_CON_DOC, S_NO_TAB_CONTAINER, S_UNKNOWN

See also GetTabName Method

Example

```
'This example gets the number and names of the tabs in the Contact view.
Dim objApp as object
Dim objViews as object
Dim objContact as object

'Initialize the Application object.
'This starts ACT! if it is not already running.
Set objApp = CreateObject("ACTOLE.APPOBJECT")

'Open the database.
objApp.OpenFile C:\My Documents\ACT\Database\ACT5demo.dbf
```

```
'Create a Views object.
Set objViews = objApp.Views

'Create the ContactView object.
Set objContact = objViews.Create(1, "MyContact")

'List all the Tabs in the ContactView.
For i = 0 To objContact.GetTabCount - 1
   List1.AddItem "TabName " & objContact.GetTabName(i)
Next i

'Close the Contact view.
objContact.Close
Set objContact = Nothing
Set objViews = Nothing
'Close the Application object.
Set objApp = Nothing
```

GetTabName Method

Requires ACT! 4.0 or later

Description Returns a string containing the name of the specified tab in the Contact view. Returns NULL on error.

Object ContactView object

Syntax *object*.**GetTabName** (*iTabNo*)

Parameters *iTabNo* A short integer that specifies the tab index number, in a range between 0 and one less than the value returned by GetTabCount.

Return type String/BSTR

GetLastError S_CON_DOC, S_NO_TAB_CONTAINER, S_UNKNOWN

See also GetTabCount Method

Example See GetTabCount.

Goto Method

Description Makes the specified contact the current contact and displays it. Returns S_ERROR on failure.

Object ContactView object

Syntax *object*.**Goto** (*szContactID*)

Parameters *szContactID* A string representing the Contact ID.

Return type Long Integer

GetLastError S_CON_DOC, S_UNKNOWN

See also BOL Method, MoveFirst Method, MoveLast Method, MoveNext Method, MovePrevious Method

GroupMembership Method

Description Returns a dispatch pointer to a grid object for the Groups tab in the Contacts view. The Groups tab must be the active tab before calling this method. Returns NULL on error.

Object ContactView object

Syntax *object*.**GroupMembership**

Return type Object/LPDISPATCH

GetLastError S_CON_DOC, S_MEM_ERROR, S_NO_DALLIST_VIEW, S_UNKNOWN

See also Activities, GetActiveTab, NotesHistory, SetActiveTab

Example

```
Set objCView = objViews.Create(1, "CV")
objCView.SetActiveTab "Groups"
'Create the Grid object for the grid on the Groups tab in the Contacts view.
Set objGM = objCView.GroupMembership
'List all the groups a contact belongs to.
For i = 0 To objGM .GetRowCount - 1
    List1.AddItem objGM .GetField(GF_Name, i)
Next i

Set objGM = Nothing
Set objCView = Nothing
```

LookupAll Method

Description Creates and displays a lookup consisting of all contact records. The result of this lookup becomes the current lookup and the previous lookup is saved.

Object ContactView object

Syntax *object.***LookupAll**

Return type Long Integer

GetLastError S_UNKNOWN, S_VW_NOT_FOUND

See also LookupFieldEx Method, LookupMyRecord Method, LookupPrevious Method, RunQuery Method

LookupFieldEx Method

Requires ACT! 4.0 or later

Description Creates and displays a contact lookup based on a specified field value. The result becomes the current lookup and the previous lookup is saved. Returns S_ERROR on failure.

Note: This method extends the functionality of LookupField in ACT! 4.0 or later.

Object ContactView object

Syntax *object.***LookupFieldEx** (*lFieldID, szFieldValue, iType*)

Parameters *lFieldID* A long integer representing the field ID of the field to be used in the lookup. See ACT! Databases for lists of field IDs and names (Field constants).

szFieldValue A string representing the field value to match. Only contacts with the specified field value in the specified field are retrieved.

iType A short integer indicating the action on the current lookup

The following table lists the values for this parameter.

Value	Setting
0	Add to the current lookup
1	Narrow the current lookup
n	Specify any other number to replace the current lookup

Return type Long Integer

GetLastError S_CON_DOC, S_INVALID_ID, S_UNKNOWN

See also LookupAll Method, LookupMyRecord Method, LookupPrevious Method, RunQuery Method

Example

```
'Example of doing a lookup from the ContactView object.
Dim objContactView As Object
Dim ContactId As String
Dim k As Integer
Dim objApp As Object

'Open the database with the default database.
Set objApp = CreateObject("ACTOLE.APPOBJECT")
'Open a database.
objApp.OpenDB C:\My Documents\ACT\Database\ACT5demo.dbf
Set objViews = objApp.Views
Set objContactView = objViews.Create(1, "CL")

'Lookup all contacts with "Cordoba" in the company field.
objContactView.LookupFieldEx CF_Company, "Cordoba", 2

'Get the count of all contacts in the lookup for error checking.
objContact.LookupAll
Reccount = objContact.GetCount
'Narrow the lookup to all contacts with a name starting with "Carl".
objContactView.LookupFieldEx CF_Name, "Carl", 1
If objContactView.GetCount = Reccount then
   MsgBox "No names starting with Carl found"
Endif

Set objContactView = Nothing
Set objViews = Nothing
Set objApp = Nothing
```

LookupMyRecord Method

Description Creates and displays a lookup consisting only of the current user's My Record. The result of this lookup becomes the current lookup and the previous lookup is saved.

Object ContactView object

Syntax *object.***LookupMyRecord**

Return type Long Integer

GetLastError S_UNKNOWN, S_VW_NOT_FOUND

See also LookupAll Method, LookupFieldEx Method, LookupPrevious Method, RunQuery Method

LookupPrevious Method

Description Returns the most recent lookup, which replaces the current lookup.

Object ContactView object

Syntax *object.***LookupPrevious**

Return type Long Integer

GetLastError S_UNKNOWN, S_VW_NOT_FOUND

See also LookupAll Method, LookupFieldEx Method, LookupMyRecord Method, RunQuery Method

MoveFirst Method

Description Displays the first record in the current lookup.

Object ContactView object

Syntax *object*.**MoveFirst**

Return type Long Integer

GetLastError S_UNKNOWN, S_VW_NOT_FOUND

See also BOL, EOL, MoveLast, MoveNext, Moveprevious

MoveLast Method

Description Displays the last record in the current lookup.

Object ContactView object

Syntax *object*.**MoveLast**

Return type Long Integer

GetLastError S_UNKNOWN, S_VW_NOT_FOUND

See also BOL Method, EOL Method, MoveFirst Method, MoveNext Method, MovePrevious Method

MoveNext Method

Description Displays the next record in the current lookup. Returns S_NONE if the next record does not exist.

Object ContactView object

Syntax *object*.**MoveNext**

Return type Long Integer

GetLastError S_UNKNOWN, S_VW_NOT_FOUND

See also BOL Method, EOL Method, MoveFirst Method, MoveLast Method, MovePrevious Method

Example

```
'This example displays the next record in the lookup.
objContact.MoveNext
If objContact.EOL then
   MsgBox "End of Contact List"
End If
```

MovePrevious Method

Description Displays the previous record in the current lookup. Returns S_NONE if the previous record does not exist.

Object ContactView object

Syntax *object*.**Moveprevious**

Return type Long Integer

GetLastError S_UNKNOWN, S_VW_NOT_FOUND

See also BOL, EOL, MoveFirst, MoveLast, MoveNext BOL Method, EOL Method, MoveFirst Method, MoveLast Method, MoveNext Method

NewContactDialog Method

Requires ACT! 4.0 or later

Description Opens the Add Contact dialog box enabling the user to enter the information for a new contact. Returns S_OK if the user clicked OK after typing the contact name and any other contact information and S_ERROR if the user clicked Cancel.

Object ContactView object

Syntax *object*.**NewContactDialog**

Return type Long Integer

GetLastError S_UNKNOWN

Comments This method will not return until the user closes the dialog box by clicking OK or Cancel.

See also AddNewContact Method

NotesHistory Method

Description Returns a dispatch pointer to a Grid object for the Notes/History tab in the Contacts view. Returns NULL on error.

Object ContactView object

Syntax *object*.**NotesHistory**

Return type Object/LPDISPATCH

GetLastError S_CON_DOC, S_MEM_ERROR, S_NO_DALLIST_VIEW, S_UNKNOWN

See also Activities Method, GetActiveTab Method, SetActiveTab Method

RunQuery Method

Description Loads and runs the specified saved query and creates and displays a lookup based on the query. Returns S_ERROR on failure.

Object ContactView object

Syntax *object*.**RunQuery** (*szQueryFile*)

Parameters szQueryFile A string representing the complete path and file name of the file containing the query to be executed.

Return type Long Integer

GetLastError S_CON_DOC, S_NOT_FOUND, S_UNKNOWN

See also LookupAll Method, LookupFieldEx Method, LookupMyRecord Method, LookupPrevious Method

Example

```
'This example loads and runs MYQUERY.QRY.
Dim objApp as object
Dim objViews as object
Dim objContact as object

'Initialize the Application object.
'This starts ACT! if it is not already running.
Set objApp = CreateObject("ACTOLE.APPOBJECT")

'Open the database.
objApp.OpenFile C:\My Documents\ACT\Database\ACT5demo.dbf

'Create a Views object.
Set objViews = App.Views
```

```
'Create the ContactView object. This brings up the Contact view.
Set objContact = objViews.Create(1, "MyContact")

'Run MyQuery. Contacts that meet the criteria are in the current lookup.
objGrp.RunQuery "C:\PROGRAM FILES\ACT\QUERY\MYQUERY.QRY"

'Close the Contact view.
objContact.Close
objViews.CloseAll
Set objContact = Nothing
Set objViews = Nothing
'Close the Application object.
Set objApp = Nothing
```

Sales Method

Requires ACT! 5.0 or later

Description Returns a dispatch pointer to the grid object for the Sales tab.

Object ContactView object

Syntax *object*.**Sales**

Return type Object/LPDISPATCH

See also CompleteSale Method, CreateSalesForecast Method

Example

```
'The following sample lists Sales records for a specified contact.
Set objViews = objApp.Views
Set objContactView = objViews.CreateEx(1, "CV", 2)
'Select the Sales/Opportunities tab.
objContactView.SetActiveTab "Sales/Opportunities"
'Go to the contact for which to list the Sales records.
objContactView.Goto ContactId
'Set the Sales Grid object.
Set objSales = objContactView.Sales
List1.AddItem objSales.GetRowCount & " Sales for this Contact"
For i = 0 To objSales.GetRowCount - 1
    List1.AddItem objSales.GetField(SLF_Status, i) & "  " &
        objSales.GetField(SLVF_ProductName, i) & "  " &
        objSales.GetField(SLVF_TypeName, i) & "  " &
        objSales.GetField(SLF_SaleDate, i)
    List1.AddItem objSales.GetField(SLF_Units, i) & "  " &
        objSales.GetField(SLF_UnitPrice, i) & "  " &
        objSales.GetField(SLF_Amount, i) & " " &
        objSales.GetField(SLVF_Competitors, i)
Next i
```

SaveQuery Method

Version ACT! 4.0 or earlier

Description Saves the current query in the specified .QRY file. Returns S_ERROR on failure.

Object ContactView object

Syntax *object*.**SaveQuery** (*szQueryFile*)

Parameters *szQueryFile* A string representing the complete path and file name of the file where the query will be saved. Query files must have a .QRY extension.

Return type Long Integer

GetLastError S_CON_DOC, S_UNKNOWN

SelectContactDlg Method

Requires ACT! 4.0 or later

Description Opens the Associate with Contact dialog box with the specified text in the Item: field to enable the user to select a contact to associate with the item. Returns the Unique ID of the selected contact if the user clicked OK or NULL if the user clicked Cancel.

Object ContactView object

Syntax *object*.**SelectContactDlg** (*szText*)

Parameters *szText* A string representing the text to be displayed in the Item: field of the Associate with Contact dialog box.

Return type String/BSTR

GetLastError S_UNKNOWN

Comments This method will not return until the user closes the dialog box by clicking OK or Cancel.

Example

```
Set objViews = objApp.Views
Set objContactView = objViews.Create(1, "CV")
ContactId = objContactView.SelectContactDlg("Select a Contact")
objContactView.Goto ContactId

'Now you can do whatever you want with this contact,
'Add a note, activity, etc.
Set objContactView = Nothing
Set objViews = Nothing
```

SetActiveGroup Method

Description Makes the group with the specified Unique ID active.

Object ContactView object

Syntax *object*.**SetActiveGroup** (*szGroupID*)

Parameters *szGroupID* A string representing the Unique ID of the group to be made active.

Note: To make <No Group> the active group, specify an empty string.

Return type Long Integer

GetLastError S_CON_DOC, S_UNKNOWN

See also GetActiveGroup Method, GetActiveGroupName Method, SetActiveGroupName Method

Example See GetActiveGroup.

SetActiveGroupName Method

Requires ACT! 4.0 or later

Description Makes the group (specified by group name) active. One group is designated as the active group in ACT! All new contacts are associated with the active group by default. Calling this method before adding a contact ensures that the new contact is associated with the specified group.

Object ContactView object

Syntax *object*.**SetActiveGroupName** (*szGroupName*)

Parameters *szGroupName*A string representing the name of the group to be made active.

Return type Long Integer

GetLastError S_CON_DOC, S_UNKNOWN

See also GetActiveGroup Method, GetActiveGroupName Method, SetActiveGroup Method

Example See GetActiveGroupName.

SetActiveTab Method

Description Makes the specified tab active. Returns S_ERROR on failure.

Object ContactView object

Syntax *object*.**SetActiveTab** (*szTabName*)

Parameters *szTabName* A string representing the name of the tab to be made active.

Return type Long Integer

GetLastError S_CON_DOC, S_INVALID_TAB, S_NO_TAB_CONTAINER, S_UNKNOWN

See also Activities Method, GetActiveTab Method, NotesHistory Method

SetField Method

Description Sets the value of the specified field in the current record. Returns S_ERROR on failure.

Object ContactView object

Syntax *object*.**SetField** (*lFieldID*, *szFieldValue*)

Parameters *lFieldID* A long integer representing the field ID of the field. See ACT! Databases for lists of field IDs and names (Field constants).

szFieldValue A string representing the value to set in the specified field.

Return type Long Integer

GetLastError S_CON_DOC, S_INVALID_INPUT, S_UNKNOWN

See also GetField Method

Example See AddNewContact Method.

TriggerActivitySeries Method

Requires ACT! 5.0 or later

Description Runs the specified activity series. Returns 0 if successful or an error code if unsuccessful. For more information, see Error codes.

Object ContactView object

Syntax *object*.**TriggerActivitySeries** (*iLookupType, SeriesDate, szActivitySeriesName*)

Parameters *iLookupType* A short integer representing the lookup type.

The following table lists the values for this parameter:

Value	Type	Value	Type
0	All records	2	Current Lookup
1	Current record	3	Selected Group

SeriesDate A date representing the start date or due date for the activity series, formatted in Windows Regional Settings Short Date style.

*szActivitySeriesName*A string specifying the file name and path for the activity series. Activity series files are stored in \ACT\Macro. The file extension must be .SER.

Return type Long Integer

Example

```
'This example code schedules the "Test.ser" series with the current lookup
'and a start date of 9/11/99.

Set objViews = objApp.Views
Set objContactView = objViews.Create(1, "CL")
ret = objContactView.TriggerActivitySeries(1, "9/11/99",
    "c:\Program Files\ACT\Macro\Test.ser")
Set objContactView = Nothing
```

ExplorerView object

The ExplorerView object provides an interface to use a Web floating view added using the methods detailed in Views and Tabs. The following methods apply only to the ExplorerView object. See Common properties and methods for properties and additional methods that apply to this object.

Sample Code

The following sample Visual Basic code demonstrates how to use the ExplorerView object:

```
'A simple example demonstrating the use of the various methods.

Dim ExplView As object

'Get the Views object
Set objViews = objApp.Views

'Load the Floating View (7), which has MyWebSite as the title.
Set ExplView = objViews.FindExplorerView("MyWebSite", 7)
If objViews.GetLastError <> 0 Then
    MsgBox "Error loading the floating Explorer View, exiting"
    objViews.ClearError
    Exit Sub
End If
List1.AddItem "View Type : (Should be 7)" & ExplView.Type
'Get the address currently open.
List1.AddItem "Get URL :" & ExplView.GetURL
'Load the URL www.actsoftware.com.
ExplView.SetURL "www.actsoftware.com"

ExplView.Close
Set ExplView = Nothing
Set objViews = Nothing
```

Methods

Name	Parameter(s)	Parameter type(s)	Return type
GetStartupURL Method	None	*	String/BSTR
GetURL Method	None	*	String/BSTR
GoBack Method	None	*	Long Integer
GoForward Method	None	*	Long Integer
Refresh Method	None	*	Long Integer
SetURL Method	szURL	String	Long Integer
Stop Method	None	*	Long Integer

GetStartupURL Method

Description Returns the URL that the Explorer uses when it is launched.

Object ExplorerView object

Syntax *object*.**GetStartupURL**

Return type String/BSTR

See also GetURL Method, SetURL Method

Example See Sample Code.

GetURL Method

Description Returns the current URL from the Explorer. Returns NULL on error.

Object ExplorerView object

Syntax *object*.**GetURL**

Return type String/BSTR

GetLastError S_CON_DOC

See also GetStartupURL Method, SetURL Method

GoBack Method

Description Displays the contents of the previous URL in the history buffer. Returns S_OK.

Object ExplorerView object

Syntax *object*.**GoBack**

Return type Long Integer

GetLastError S_CON_DOC

See also GoForward Method

GoForward Method

Description Displays the contents of the next URL in the history buffer.

Object ExplorerView object

Syntax *object*.**GoForward**

Return type Long Integer

GetLastError S_CON_DOC

See also GoBack Method

Refresh Method

Description Refreshes the contents of the Explorer view using the current URL.

Object ExplorerView object

Syntax *object*.**Refresh**

Return type Long Integer

GetLastError S_CON_DOC

SetURL Method

Description Sets the specified URL as the current URL for the Explorer.

Object ExplorerView object

Syntax *object*.**SetURL** (*szURL*)

Parameters *szURL* A string representing the new URL.

Return type Long Integer

GetLastError S_CON_DOC

See also GetURL Method

Stop Method

Description Stops the Explorer from retrieving the contents of the current URL.

Object ExplorerView object

Syntax *object*.**Stop**

Return type Long Integer

GetLastError S_CON_DOC

Grid object

The Grid object provides an interface to all ACT! grid functionality, including the Task List view, the Contact List View, and the contact tabs Activities, Groups, Notes/Histories, and Sales. Use the this object to retrieve data from any of the listed views. The following methods apply only to the Grid object.

Methods

Name	Parameter(s)	Parameter type(s)	Return type
BOL Method	None	*	Boolean
DeleteRow Method	lRowNo	Long Integer	Long Integer
EOL Method	None	*	Boolean
GetColumnCount Method	None	*	Long Integer
GetColumnID Method	lColNo	Long Integer	Long Integer
GetColumnName Method	lColNo	Long Integer	String/BSTR
GetCurrentRow Method	None	*	Long Integer
GetField Method	lFieldID,lRowNo	Long Integer, Long Integer	String/BSTR

Name	Parameter(s)	Parameter type(s)	Return type
GetFilter Method	lFilters vDate vUserID	Long Integer Variant Variant	Long Integer
GetLastError Method	None	*	Long Integer
GetRowCount Method	None	*	Long Integer
GetRowNumber Method	szUniqueID	String	Long Integer
GetUniqueID Method	lRowNo	Long Integer	String/BSTR
Goto Method	lRowNo	Long Integer	Long Integer
MoveFirst Method	None	*	Long Integer
MoveLast Method	None	*	Long Integer
MoveNext Method	None	*	Long Integer
MovePrevious Method	None	*	Long Integer
RefreshGrid Method	None	*	Long Integer
SelectRow Method	lRowNo,bReserved	Long Integer,Boolean	Long Integer
SetField Method	lFieldID,lRowNo,szFieldValue	Long Integer,Long Integer,String	Long Integer
SetFilter Method	lFilters, szDate,szUserID…	Long IntegerString, String	Long Integer
Sort Method	lFieldID,True|False	Long Integer, Boolean	Long Integer

BOL Method

Description Returns True if the current record is the first in the grid or False if it is not the first in the grid or on error.

Object Grid object

Syntax *object*.**BOL**

Return type Boolean

GetLastError S_NO_DALLIST_VIEW, -2

See also EOL Method, Goto Method, MoveFirst Method, MoveLast Method, MoveNext Method, MovePrevious Method

Example

```
Dim objApp as object
Dim objContactListView as object
Dim objContactListGrid as object

'Initialize the Application object.
'This starts ACT! if it is not already running.
Set objApp = CreateObject("ACTOLE.APPOBJECT")

'Open the database.
objApp.OpenFile C:\My Documents\ACT\Database\ACT5demo.dbf

Set objViews = objApp.Views

'Create the Contact List view.
Set objContactListView = objViews.Create(2, "CL")
```

```
'Create the Grid object for the Contact List.
Set objContactListGrid = objContactListView.GetGrid
If objContactListGrid.<> True Then
   objContactListGrid.MoveFirst
Else
   MsgBox "Beginning of List"
End If

Set objContacttListGrid = Nothing
objContacttListView.Close
Set objCtListView = Nothing
Set objViews = Nothing
'Close the Application object.
Set objApp = Nothing
```

DeleteRow Method

Description Deletes the specified record from the grid and table. Use the SelectRow method or go to the row to be deleted before using this method. This method cannot be used on Grid object for Group Membership tab (Contact view) or Contact tab (Group view). Returns S_ERROR on failure.

Object Grid object

Syntax *object*.**DeleteRow** (*lRowNo*)

Parameters *lRowNo* A long integer representing the row of the record to be deleted, in the range between 0 (zero) and one less than the value of the row count.

Return type Long Integer

GetLastError S_DELETE_FAIL, S_DELETE_NOTALLOWED, S_INVALID_INPUT, S_NO_DALLIST_VIEW, -2

See also GetRowCount Method

EOL Method

Description Returns True if the current record is at the last in the grid and False if it is not the last in the grid or on error.

Object Grid object

Syntax *object*.**EOL**

Return type Boolean

GetLastError S_NO_DALLIST_VIEW, -2

See also BOL Method, MoveFirst Method, MoveLast Method, MoveNext Method, MovePrevious Method

Example

```
Dim objApp as object
Dim objContactListView as object
Dim objContactListGrid as object

'Initialize the Application object.
'This starts ACT! if it is not already running.
Set objApp = CreateObject("ACTOLE.APPOBJECT")

'Open the database.
objApp.OpenFile C:\My Documents\ACT\Database\ACT5demo.dbf

Set objViews = objApp.Views
```

```
'Create the Contact List view.
Set objContactListView = objViews.Create(2, "CL")

'Create the Grid object for the Contact List.
Set objContactListGrid = objContactListView.GetGrid
If objContactListGrid.EOL <> True Then
   objContactListGrid.MoveLast
Else
   MsgBox "End of List"
End If

Set objContacttListGrid = Nothing
objContacttListView.Close
Set objCtListView = Nothing
Set objViews = Nothing
'Close the Application object.
Set objApp = Nothing
```

GetColumnCount Method

Description Returns the total number of columns in the grid. Returns -1 on failure. Use GetLastError to get information on an error.

Object Grid object

Syntax *object*.**GetColumnCount**

Return type Long Integer

GetLastError S_NO_DALLIST_VIEW

Example

```
'Get the number of rows and columns in a list and
'then list all the Column IDs and column names
Dim objApp as object
Dim objTask as object
Dim objTaskGrid as object

'Initialize the Application object.
'This starts ACT! if it is not already running.
Set objApp = CreateObject("ACTOLE.APPOBJECT")

'Open the database.
objApp.OpenFile C:\My Documents\ACT\Database\ACT5demo.dbf

'Initialize the Views object.
Set objViews = objApp.Views

'Create a Task List.
Set objTask = objViews.Create(4, "TL")

'Create the Task List view Grid Object.
Set objTaskGrid = objTask.GetGrid

'Get the number of rows and columns.
List1.AddItem "There are " & objTaskGrid.GetRowCount & " Tasks"
List1.AddItem "There are " & objTaskGrid.GetColumnCount &
   " Columns in the Task List"
```

```
'List all the columns and their names.
For i = 1 To objTaskGrid.GetColumnCount
    List1.AddItem objTaskGrid.GetColumnId(i) & " : " &
        objTaskGrid.GetColumnName(i)
Next i

List1.AddItem "Closing Task List view"
objTask.Close
Set objViews = Nothing
'Close the Application object.
Set objApp = Nothing
```

GetColumnID Method

Description Returns the Field ID of the column in the grid. Returns -1 on failure. Use GetLastError to get information on an error.

Object Grid object

Syntax *object*.**GetColumnID** (*lColNo*)

Parameters lColNo A long integer representing the column of the record in the grid, in a range between 0 and one less than the value of GetColumnCount.

Return type Long Integer

GetLastError S_NO_DALLIST_VIEW, S_INVALID_INPUT

Example

```
Dim objCView As Object
Dim objViews As Object
Dim objTask As Object
Dim objTaskGrid As Object
Dim i As Integer
Dim iColumnCount As Integer
Dim objApp As Object

'Open the database with the default database.
Set objApp = CreateObject("ACTOLE.APPOBJECT")
'Open a database.
objApp.OpenDB C:\My Documents\ACT\Database\ACT5demo.dbf
Set objViews = objApp.Views

'Create a task list.
Set objTask = objViews.Create(4, "TL")
'Created the TaskView.
Set objTaskGrid = objTask.GetGrid
iColumnCount = objTaskGrid.GetColumnCount
For i=0 to iColumnCount
    MsgBox "Column ID: " & objTaskGrid.GetColumnID(i) & " has name: "
        & objTaskGrid.GetColumnName(i)
Next i
objTask.Close

Set objViews = Nothing
Set objApp = Nothing
```

GetColumnName Method

Description Returns the name of the column in the grid. Returns a blank string or NULL on error.

Object Grid object

Syntax *object*.**GetColumnName** (*lColNo*)

Parameters *lColNo* A long integer representing the name of the column in the grid, in a range between 0 and one less than the value of GetColumnCount.

Return type String/BSTR

GetLastError S_NO_DALLIST_VIEW

Example See GetColumnCount Method.

GetCurrentRow Method

Description If editing a record, returns the 0-based integer of the visible records. If a row is selected, returns the 0-based index of the row currently being edited in the grid. The row number is in a range between 0 and one less than the value of GetRowCount. Returns -1 if no row is currently being edited and on failure. Use GetLastError to get information on an error.

Object Grid object

Syntax *object*.**GetCurrentRow**

Return type Long Integer

GetLastError S_NO_DALLIST_VIEW, -2

GetField Method

Description Gets the value of the specified field in the specified row. Returns a blank string or NULL if the specified field does not exist, and on failure. Use GetLastError to get information on an error.

Object Grid object

Syntax *object*.**GetField** (*lFieldID*, *lRowNo*)

Parameters *lFieldID* A long integer representing the field ID of the field whose value is to be retrieved. See ACT! Databases for lists of field IDs and names (Field constants).

lRowNo A long integer representing the row number of the field, in a range between 0 and one less than the value of GetRowCount.

Return type String/BSTR

GetLastError S_INVALID_ID, S_NO_DALLIST_VIEW, S_INVALID_INPUT, S_GETFIELD_FAIL

See also SetField Method

Example See GetGrid Method in the ContactListView object.

GetFilter Method

Requires ACT! 4.0 or later

Description Gets the settings of the filters on the Notes/History or Activities tab of the Contacts or Groups view, or theTask List view. Can also be used to return the filter settings for the Sales Opportunity tab on the Contact view. First, define empty variables, then this method returns values representing settings of the filters. Returns S_ERROR on failure.

Caution: In Visual C++ you need to initialize vDate and vUserID before using this method.

Object Grid object

Syntax *object*.**GetFilter** (*lFilters, vDate, vUserID*)

Parameters

lFilters A long integer type variable containing the settings of filters on the Notes/History tab, Activities tab, or Task List view. After you define lFilters, a total is returned that represents all selected filters.

The lFilters total for the Notes/History tab includes the following values:

Value	Option selected	Value	Option selected
1	Show data for All Users	8	Show Attachments
2	Show Notes	16	Show E-mail (ACT! 5.0 or later)
4	Show Histories		

The lFilters total for the Activities tab or the Task List view includes the following values:

Value	Option selected	Value	Option selected
1	Show data for All Users	32	Show High priority activities
2	Show Cleared Activities	64	Show Medium priority activities
4	Show Calls	128	Show Low priority activities
8	Show Meetings	256	Show Only Timeless activities
16	Show To-do's	512	Show Outlook activities Requires ACT! 5.0 or later and Outlook)

The lFilters total for the Sales tab includes the following values:.

Value	Option selected	Value	Option selected
1	Show data for All Users	4	Show Closed/Won
2	Show Opportunities	8	Show Lost

vDate A variant (pointer to VARIANT in Visual C++) type variable containing the setting of the DatesTo Show filter, returned in the following format:

`Type\StartDate\EndDate`

The following table lists the values returned for the Type field. For types 1 to 5 the StartDate and EndDate are not returned. For type 6 StartDate and EndDate are returned in mm/dd/yy format.

Value	Option selected	Value	Option selected
1	Show All	4	Show Today
2	Show Past	5	Show Tomorrow(not supported)
3	Show Today And Future	6	Show Date Range

vUserID A variant (pointer to VARIANT in Visual C++) type variable containing the Unique ID(s) of users selected for the filter. The values are returned in the following format:

`ID1ID2ID3…`

Note: Multiple Unique IDs are returned as a continuous string of 12 character values, with no delimiters between the Unique ID values.

Return type Long Integer

GetLastError S_INVALID_ID, S_NO_DALLIST_VIEW, S_INVALID_INPUT, S_GETFIELD_FAIL

See also SetField Method, SetFilter Method

Example

```
Dim x as Long
Set objApp = CreateObject("ACTOLE.APPOBJECT")
Set objViews = objApp.Views
Set objCView = objViews.Create(1, "CV")
objCView.SetActiveTab "Notes/History"
Set objNH = objCView.NotesHistory

'Get the current filter settings.
objNH.GetFilter x, y, z
j = 0 Or 1 Or 2 Or 8 Or 4

'Set filter to select all notes, histories, e-mails, and attachments.
'Dates to Show is All and all users are selected.
objNH.SetFilter j, 1, " "

'Add the note and get the Unique ID.
uid = objCView.AddNoteHistoryEx(100)

'Get the row number.
nRow = objNH.GetRowNumber(uid)

'Set the regarding text.
objNH.SetField NHF_Text, i, "I am adding a test note"

'Reset the filter settings back to the previous settings.
objNH.SetFilter x, y, z

Set objNH = Nothing
Set objCView = Nothing
Set objViews = Nothing
```

GetLastError Method

Description Returns a long integer representing the last error code for the object. For more information, see Error codes.

Object Grid object

Syntax object.GetLastError

Return type Long Integer

GetRowCount Method

Description Returns the total number of records or rows in the grid. Returns -1 on failure. Use GetLastError to get information on an error.

Note: The total returned by this method is the total number of records that have been processed for the grid when the method was used. You may need to add a pause to get a complete total.

Object Grid object

Syntax *object*.**GetRowCount**

Return type Long Integer

GetLastError S_NO_DALLIST_VIEW

Example See GetColumnCount Method.

GetRowNumber Method

Description Returns the row number of the specified record in the grid. The row number is in a range between 0 and one less than the value of GetRowCount. Returns -1 if the row is not visible in the window or the specified row does not exist, and on failure. Use GetLastError to get information on an error.

Object Grid object

Syntax *object*.**GetRowNumber** (*szUniqueID*)

Parameters szUniqueID A string representing the Unique ID of the row.

Return type Long Integer

GetLastError -2

Example See GetColumnCount Method; AddNewActivityEx Method in ContactView object

GetUniqueID Method

Description Returns the Unique ID of the specified row in the grid.

Object Grid object

Syntax *object*.**GetUniqueID** (*lRowNo*)

Parameters *lRowNo* A long integer representing the row number of the row, in a range between 0 and one less than the value of GetRowCount.

Return type String/BSTR

GetLastError S_INVALID_INPUT, S_NO_DALLIST_VIEW, -2

Example See AddNewActivityEx Method in ContactView object.

Goto Method

Description Makes the specified record the current record.

Object Grid object

Syntax *object*.**Goto** (*lRowNo*)

Parameters lRowNo A long integer representing the row number of the row, in a range between 0 and one less than the value of GetRowCount.

Return type Long Integer

GetLastError S_NO_DALLIST_VIEW, -2

See also MoveFirst Method, MoveLast Method, MoveNext Method, MovePrevious Method

MoveFirst Method

Description Displays the first record in the grid.

Object Grid object

Syntax *object*.**MoveFirst**

Return type Long Integer

GetLastError S_NO_DALLIST_VIEW, -2

See also BOL Method, EOL Method, MoveLast Method, MoveNext Method, MovePrevious Methods

Example See BOL.

MoveLast Method

Description Displays the last record in the grid.

Object Grid object

Syntax *object*.**MoveLast**

Return type Long Integer

GetLastError S_NO_DALLIST_VIEW, -2

See also BOL Method, EOL Method, MoveFirst Method, MoveNext Method, MovePrevious Methods

Example See EOL.

MoveNext Method

Description Displays the next record in the grid. Returns S_NONE if the next record does not exist.

Object Grid object

Syntax *object*.**MoveNext**

Return type Long Integer

GetLastError S_NO_DALLIST_VIEW, -2

See also BOL Method, EOL Method, MoveFirst Method, MoveLast Method, MovePrevious Methods

MovePrevious Method

Description Displays the previous record in the grid. Returns S_NONE if the previous record does not exist.

Object Grid object

Syntax *object*.**MovePrevious**

Return type Long Integer

GetLastError S_NO_DALLIST_VIEW, -2

See also BOL Method, EOL Method, MoveFirst Method, MoveLast Method, MoveNext Methods

RefreshGrid Method

Description Refreshes the contents of the grid.

Object Grid object

Syntax *object*.**RefreshGrid**

Return type Long Integer

GetLastError -2

SelectRow Method

Description Selects (highlights) the specified row. Returns S_ERROR on failure or if the specified row does not exist. Use GetLastError to distinguish between the two cases.

Object Grid object

Syntax *object*.**SelectRow** (*lRowNo, bReserved*)

Parameters

lRowNo	A long integer representing the row number of the field.
bReserved	A Boolean value reserved for future use.

Return type Long Integer

GetLastError S_NO_DALLIST_VIEW, -2

See also Goto Method

SetField Method

Description Sets the value of the specified field in the specified row. Returns S_ERROR on failure or if the specified field does not exist. Use GetLastError to distinguish between the two cases.

Object Grid object

Syntax *object*.**SetField** *(lFieldID, lRowNo, szFieldValue*)

Parameters *lFieldID* A long integer representing the field ID of the field whose value is to be set. See ACT! Databases for lists of field IDs and names (Field constants).

lRowNo A long integer representing the row number of the field, in a range between 0 and one less than the value of RowCount.

szFieldValue A string containing the field value.

Return type Long Integer

GetLastError S_INVALID_ID, S_INVALID_INPUT, S_NO_DALLIST_VIEW, S_SET_NOT_ALLOWED, S_SETFIELD_FAIL, S_STOP_EDIT, -2

See also GetField Method

Example See AddNewActivityEx Method in ContactView and AddNewActivityEx Methodthe TaskListView objects, AddNoteHistoryEx Method in ContactView object, AddNoteEx Method in GroupView object, AddNewContactEx Method in ContactListView object.

SetFilter Method

Requires ACT! 4.0 or later

Description Sets the filters on the Notes/History or Activities tab of the Contacts or Groups view, or the Task List view. This should be done before adding records into the grid. This method is recommended to avoid incorrect information getting added to newly added Notes/History and Activity records. Returns S_ERROR on failure.

Object Grid object

Syntax *object*.**SetFilter** (*lFilters, szDate, szUserID…*)

Parameters *lFilters* A long integer obtained by ORing 0 with the values for filters on the Notes/History tab, Activities tab, or Task List view that you want selected.

The following table lists the values that are included in the total for lFilters for the Notes/History tab.

Value	Option selected	Value	Option selected
1	Show data for All Users	8	Show Attachments
2	Show Notes	16	Show E-mail (ACT! 5.0)
4	Show Histories		

The following table lists the values that are included in the total for lFilters for the Activities tab or the Task List view.

Value	Option selected	Value	Option selected
1	Show data for All Users	32	Show High priority activities
2	Show Cleared Activities	64	Show Medium priority activities
4	Show Calls	128	Show Low priority activities
8	Show Meetings	256	Show Only Timeless activities
16	Show To-do's	512	Show Outlook activities (ACT! 5.0 or later and Outlook)

szDate A string representing the dates for the items to be filtered, specified in the format:

`Type\StartDate\EndDate`

The following table lists values for the Type field. For types 1 to 5 the StartDate and EndDate fields are omitted. For type 6, specify the StartDate and EndDate in mm/dd/yy format.

Value	Option selected	Value	Option selected
1	Show All	4	Show Today
2	Show Past	5	Show Tomorrow(not supported)
3	Show Today And Future	6	Show Date Range

szUserID A string representing the Unique ID(s) of users specified for the filter. Specify szUserID in the following format:

`ID1\ID2\ID3…`

Note: Specify multiple Unique IDs as a continuous string of 12?character values, with no delimiters between the Unique ID values.

Return type Long Integer

GetLastError S_INVALID_INPUT, S_NO_DALLIST_VIEW

See also GetField Method

Example See GetFilter Method.

Sort Method

Description Sorts the display grid by the specified column.

Object Grid object

Syntax *object*.**Sort** (*lFieldID, True|False*)

Parameters *lFieldID* A long integer representing the field ID of the column in the grid to be used for the sort. See ACT! Databases for lists of field IDs and names (Field constants).

True|False Specify True to sort the specified column in descending order or False to sort it in ascending order.

Return type Long Integer

GetLastError S_BUILD_ERROR, S_NO_DALLIST_VIEW, S_NO_SCHEMA, S_NOT_SORTABLE, -2

Example See AddNewActivityEx Method in TaskListView object.

GroupView object

The GroupView object provides an interface to use the Group view in the ACT! application. The following methods apply only to the GroupView object. See Common properties and methods for properties and additional methods that apply to this object.

Methods

Name	Parameter(s)	Parameter type(s)	Return type
Activities Method	None	*	Object/LPDISPATCH
AddMemberToGroup Method	szContactID	String	Long Integer
AddNew Method	None	*	Long Integer

Name	Parameter(s)	Parameter type(s)	Return type
AddNewSubGroup Method	szUniqueID	String	Long Integer
AddNoteEx Method	None	*	String/BSTR
AttachFile Method	szFilename	String	String/BSTR
BOL Method	None	*	Boolean
ChangeToParentGroup Method	None	*	Long Integer
ChangeToSubGroup Method	szUniqueID	String	Long Integer
Collapse Method	None	*	Long Integer
ContactMembers Method	None	*	Object/LPDISPATCH
Delete Method	None	*	Long Integer
DeleteGroupFast Method	None	*	Long Integer
EOL Method	None	*	Boolean
Expand Method	None	*	Long Integer
GetActiveTab Method	None	*	String/BSTR
GetCount Method	None	*	Short Integer
GetCurrentID Method	None	*	String/BSTR
GetField Method	lFieldID	Long Integer	String/BSTR
GetSubGroupCount Method	None	*	Long Integer
GetTabCount Method	None	*	Short Integer
GetTabName Method	iTabNo	Short Integer	String/BSTR
Goto Method	szGroupID	String	Long Integer
GroupType Method	None	*	Short Integer
IsExpanded Method	None	*	Boolean
LookupAll Method	None	*	Long Integer
LookupFieldEx Method	lFieldID szFieldValue iType	Long Integer String Short Integer	Long Integer
LookupPrevious Method	None	*	Long Integer
MoveFirst Method	None	*	Long Integer
MoveLast Method	None	*	Long Integer
MoveNext Method	None	*	Long Integer
MovePrevious Method	None	*	Long Integer
NotesHistory Method	None	*	Object/LPDISPATCH
RunQuery Method	szQueryFile	String	Long Integer
SaveQuery Method	szQueryFile	String	Long Integer
SetActiveTab Method	szTabName	String	Long Integer
SetField Method	lFieldID szFieldValue	Long Integer String	Long Integer

Activities Method

Description Returns a dispatch pointer to a Grid object for the Activities tab. It does not make the Activities tab the active tab. The Activities tab must be the active tab before calling this method. Returns NULL on error.

Object GroupView object

Syntax *object.***Activities**

Return type Object/LPDISPATCH

GetLastError S_NO_DALLIST_VIEW, S_MEM_ERROR, S_CON_DOC, S_FRM_VW_SYS

See also GetActiveTab Method, NotesHistory Method, SetActiveTab Method

Example

```
'This example opens the Activities tab and returns the
'number of activities for the group.

Dim objGrp as object
Dim objActivities as object
Dim objContacts as object

Set objViews = objApp.Views

'Open the database.
objApp.OpenFile C:\My Documents\ACT\Database\ACT5demo.dbf

Set objGrp = objViews.Create(3, "Group")

'Set Activities to be the active tab.
objGrp.SetActiveTab "Activities"
objGrp.MoveFirst
Set objActivities = objGrp.Activities

'Print the row count, which is the same as the number of
'activities for the first group.
List1.AddItem "This group has " & objActivities.GetRowCount & " activities"
Set objActivities = Nothing
objGrp.Close
Set objGrp = Nothing

'Close the application.
Set objActivities = Nothing
Set objApp = Nothing
```

AddMemberToGroup Method

Description Makes the specified contact as a member of the current group. The specified contact will appear in the Contacts tab of the Group view. This function fails if the specified contact is an existing member of the current group. Returns S_ERROR on failure.

Object GroupView object

Syntax *object.***AddMemberToGroup** (*szContactID*)

Parameters *szContactID* A string representing the ID of the contact to be added.

Return type Long Integer

GetLastError S_INVALID_ID, S_ADD_CONTACT, S_DUPLICATE_CONTACT

See also AddContactToGroup Method in ContactView object

Example

```
'This example adds sContact to the current group.
Dim objApp as object
Dim objGrp as object
Dim objGrid as object
Dim sContact as string

'Initialize the Application object.
'This starts ACT! if it is not already running.
Set objApp = CreateObject("ACTOLE.APPOBJECT")

'Open the database.
objApp.OpenFile C:\My Documents\ACT\Database\ACT5demo.dbf

'Create a Group view.
Set objViews = objApp.Views
Set objGrp = objViews.Create(3, "Group")

'Set the active tab to Contacts.
objGrp.SetActiveTab "Contacts"

Set objGrid = objGrp.ContactMembers
objGrp.AddMemberToGroup (sContact)

objGrp.Close
Set objGrp = Nothing

'Close the Application object.
Set objApp = Nothing
```

AddNew Method

Description Displays a new group with all fields blank. Return S_ERROR on failure.

Object GroupView object

Syntax *object.***AddNew**

Return type Long Integer

GetLastError S_MAIN_WND

See also Delete Method

Example

```
'This example displays a new group.
Dim objApp as object
Dim objViews as object
Dim objGroup as object

'Initialize the Application object.
'This starts ACT! if it is not already running.
Set objApp = CreateObject("ACTOLE.APPOBJECT")

'Open the database.
objApp.OpenFile C:\My Documents\ACT\Database\ACT5demo.dbf

'Create a Views object.
Set objViews = App.Views
```

```
'Create the GroupView object. This brings up the Groups view.
Set objGroup = objViews.Create(3, "MyGroup")

'Bring up the "Add New Group View".
'Add the new group "My group".
objGroup.AddNew
objGroup.SetField GF.Name, "My group"
objGroup.SetField GF.Division, "Division"
objGroup.SetField GF.Address1, "Address 1"
objGroup.SetField GF.City, "City"
objGroup.SetField GF.State, "State"

'Close the Group view.
objGroup.Close
objViews.CloseAll
Set objGroup = Nothing
Set objViews = Nothing
'Close the Application object.
Set objApp = Nothing
```

AddNewSubGroup Method

Requires ACT! 5.0 or later

Description Creates a subgroup for the specified parent group with all blank fields. Returns 0 if successful or returns an error code if unsuccessful. For more information, see Error codes.

Object GroupView object

Syntax *object*.**AddNewSubGroup** (*szUniqueID*)

Parameters *szUniqueID* A string that represents the Unique ID of the parent group record for which you want to create the subgroup.

Return type Long Integer

See also ChangeToParentGroup Method, ChangeToSubGroup Method, Collapse Method, Expand Method, GetSubGroupCount Method, GroupType Method, IsExpanded Method

Example

```
'This example code adds a new subgroup to the first group in the Groups view.
Set objViews = objApp.Views
Set objGrp = objViews.Create(3, "Group")
objGrp.MoveFirst
'Get the Unique ID of the first group in the list.
uid = objGrp.GetCurrentID
'Add a new subgroup to the first group.
objGrp.AddNewSubGroup uid
'Populate the group's details.
objGrp.SetField GF_Name, "Test Subgroup"
objGrp.SetField GF_Division, "Division : SDK"
objGrp.SetField GF_Region, "Western"
objGrp.SetField GF_Employees, "120"
```

AddNoteEx Method

Requires ACT! 4.0 or later

Description Adds a note in the Notes/History tab of the Groups view. Use SetField in the Grid object to set other fields in the newly created row. Returns a string containing the Unique ID of the note. Use GetFilter and SetFilter in the Grid object before using this method.

Note: This method is recommended to be used instead of AddNote in ACT! 4.0 or later.

Object GroupView object

Syntax *object*.**AddNoteEx**

Return type String/BSTR

GetLastError S_MAIN_WND, S_NO_DALLIST_VIEW, S_MEM_ERROR

See also AddNoteEx Method

AddNoteHistoryEx Method in ContactView object

GetField Method, SetField Method in Grid object

Example

```
'This example adds a note to the current group.
Set objViews = objApp.Views
'Create the GroupView object.
Set objGrp = objViews.Create(3, "Group")

objGrp.SetActiveTab "Notes/History"
'Create the grid object in the Notes/History tab of the Groups view.
Set objNH = objGrp.NotesHistory

Dim uid As String
'Call AddNoteEx and get the Unique ID of the newly added record.
uid = objGrp.AddNoteEx

'Get the row number for the Unique ID.
i = objNH.GetRowNumber(uid)

'Set the Regarding text of the note.
objNH.SetField NHF_Text, i, "Group-Note Test1"
i = objNH.GetRowNumber(uid)

'Set the Date/Time of the note.
objNH.SetField NHF_UserTime, i, "1/1/98 8:00AM"

Set objNH = Nothing
objGrp.Close
Set objGrp = Nothing
```

AttachFile Method

Requires ACT! 4.0 or later

Description Adds the specified file as an attachment to the Notes/History tab of the Groups view. Returns the Unique ID of the new Notes/History record for the attachment on success and an empty string on failure.

Object GroupView object

Syntax *object*.**AttachFile** (*szFilename*)

Parameters *szFilename* A string representing the complete path and the file name of the file to add as an attachment.

Return type String/BSTR

GetLastError S_CON_DOC, S_NOT_FOUND, S_ERROR

See also AttachFile Method in ContactView object

Example

```
'This example attaches a file (an attachment record) in the
'Notes/History tab of the current record.

Set objViews = objApp.Views
'Create the GroupView object.
Set objGrp = objViews.Create(3, "Group")

objGrp.SetActiveTab "Notes/History"
'Create the Grid object in the Notes/History tab of the Groups view.
Set objNH = objGrp.NotesHistory

'Create an attachment record in the Notes/History tab for the
'current view and attach the file.
objGrp.AttachFile "c:\My Documents\features.doc"

Set objNH = Nothing
objGrp.Close
Set objGrp = Nothing
```

BOL Method

Description Indicates whether the current group is the first in the lookup (beginning of lookup). Returns True if the current group is at the beginning of the lookup or False if it is not at the beginning of lookup or on failure. Use GetLastError to distinguish between the two cases.

Object GroupView object

Syntax *object*.**BOL**

Return type Boolean

GetLastError S_MAIN_WND, S_CON_DOC

See also EOL Method, Goto Method, MoveFirst Method, MoveLast Method, MoveNext Method, MovePrevious Method

ChangeToParentGroup Method

Requires ACT! 5.0 or later

Description Changes the current subgroup to a parent group. Returns 0 if successful or an error code if unsuccessful. For more information, see Error codes.

Object GroupView object

Syntax *object*.**ChangeToParentGroup**

Return type Long Integer

See also AddNewSubGroup Method, ChangeToSubGroup Method, Collapse Method, Expand Method, GetSubGroupCount Method, GroupType Method, IsExpanded Method

Example

```
'This example code changes the current subgroup to a parent group.
Set objViews = objApp.Views
Set objGrp = objViews.CreateEx(3, "GV",1)
objGrp.Goto guid

'If the group type is subgroup, then change it to a parent group.
If objGrp.GroupType = 0 Then
   objGrp.ChangeToParentGroup
End If
```

ChangeToSubGroup Method

Requires ACT! 5.0 or later

Description Changes the current group or subgroup to a subgroup of the specified parent group. Returns 0 if successful or an error code if unsuccessful. For more information, see Error codes.

Object GroupView object

Syntax *object*.**ChangeToSubGroup** (*szUniqueID*)

Parameters *szUniqueID* A string that specifies the Unique ID of the parent group record.

Return type Long Integer

See also AddNewSubGroup Method, ChangeToParentGroup Method, Collapse Method, Expand Method, GetSubGroupCount Method, GroupType Method, IsExpanded Method

Collapse Method

Requires ACT! 5.0 or later

Description Collapses the current node of the tree so that the subgroups of the current group are not visible.

Object GroupView object

Syntax *object*.**Collapse**

Return type Long Integer

See also AddNewSubGroup Method, ChangeToParentGroup Method, ChangeToSubGroup Method, Expand Method, GetSubGroupCount Method, GroupType Method, IsExpanded Method

Example

```
Set objViews = objApp.Views
Set objGrp = objViews.Create(3, "Group")

objGrp.MoveFirst
While objGrp.EOL <> True
    If objGrp.IsExpanded = False Then
    'Tree for that particular group is collapsed
        objGrp.Expand
    Else
        objGrp.Collapse
    End If
objGrp.MoveNext
Wend
```

Example

```
'This sample code goes through a group list.
Set objViews = objApp.Views
Set objGrp = objViews.Create(3, "Group")

objGrp.MoveFirst
list1.AddItem "First Group is " & objGrp.GetField(GF_Name)
While objGrp.EOL <> True
    If objGrp.GroupType = 0 Then
        list1.AddItem objGrp.GetField(GF_Name) & " is a parent"
        objGrp.Expand
        For i = 1 To objGrp.GetSubGroupCount
            objGrp.MoveNext
            list1.AddItem " Sec group: " & objGrp.GetField(GF_Name)
        Next i
```

```
        Else
            list1.AddItem objGrp.GetField(GF_Name) & " is a sub group"
        End If
    objGrp.MoveNext
    Wend
```

ContactMembers Method

Description Returns a dispatch pointer to a grid object for the Contacts tab in the Groups view. Returns NULL on error.

Object GroupView object

Syntax *object.***ContactMembers**

Return type Object/LPDISPATCH

GetLastError S_NO_DALLIST_VIEW, S_MEM_ERROR, S_CON_DOC, S_FRM_VW_SYS

See also Activities Method, GetActiveTab Method, NotesHistory Method, SetActiveTab Method

Example See AddMemberToGroup Method.

Delete Method

Description Deletes the active (selected) group and all details about the group including activities, notes, and histories. Displays a dialog box to confirm the deletion. The contacts belonging to the group are not deleted. Returns S_ERROR on failure.

Object GroupView object

Syntax *object.***Delete**

Return type Long Integer

GetLastError S_MAIN_WND

See also AddNew Method

DeleteGroupFast Method

Requires ACT! 5.0 or later

Description Deletes the current group without invoking a confirmation dialog box. Returns 0 if successful or an error code if unsuccessful. For more information, see Error codes.

Object GroupView object

Syntax *object.***DeleteGroupFast**

Return type Long Integer

See also DeleteContactFast Method in ContactView object

EOL Method

Description Indicates whether the current group is the last in the lookup (end of lookup). Returns True if the current group is at the end of the lookup or False if it is not at the end of lookup or on failure. Use GetLastError to distinguish between the two cases.

Object GroupView object

Syntax *object.***EOL**

Return type Boolean

GetLastError S_MAIN_WND, S_CON_DOC

See also BOL Method, Goto Method, MoveFirst Method, MoveLast Method, MoveNext Method, MovePrevious Method

Expand Method

Requires ACT! 5.0 or later

Description Expands the current node of the tree so the subgroups of the current group are visible. Returns 0 if successful or an error code if unsuccessful. For more information, see Error codes.

Object GroupView object

Syntax *object*.**Expand**

Return type Long Integer

See also AddNewSubGroup Method, ChangeToParentGroup Method, Collapse Method, ChangeToSubGroup Method, GetSubGroupCount Method, GroupType Method, IsExpanded Method

Example

```
Set objViews = objApp.Views
Set objGrp = objViews.Create(3, "Group")

objGrp.MoveFirst
While objGrp.EOL <> True
    If objGrp.IsExpanded = False Then
    'Tree for that particular group is collapsed.
        objGrp.Expand
    Else
        objGrp.Collapse
    End If
objGrp.MoveNext
Wend
```

GetActiveTab Method

Description Returns the name of the currently active tab in the Groups view.

Object GroupView object

Syntax *object*.**GetActiveTab**

Return type String/BSTR

GetLastError S_CON_DOC, S_NO_TAB_CONTAINER

See also Activities Method, NotesHistory Method, SetActiveTab Method

Example

```
'This example gets the name of the active tab.
Dim objApp as object
Dim objViews as object
Dim objGroup as object
Dim i as integer

'Initialize the Application object.
'This starts ACT! if it is not already running.
Set objApp = CreateObject("ACTOLE.APPOBJECT")

'Open the database.
objApp.OpenFile C:\My Documents\ACT\Database\ACT5demo.dbf

'Create a Views object.
Set objViews = App.Views
```

```
'Create the GroupView object. This brings up the GroupView.
Set objGroup = objViews.Create(3, "MyGroup") Dim objApp as object

'If the active tab is not Activities, set the active tab to Activities.
If objGrp.GetActiveTab <> "Activities" Then
   objGrp.SetActiveTab "Activities"
End If

'Close the Groups view.
objGroup.Close
objViews.CloseAll
Set objGroup = Nothing
Set objViews = Nothing
'Close the Application object.
Set objApp = Nothing
```

GetCount Method

Description Returns the number of group records in the current lookup. Returns -1 on failure.

Object GroupView object

Syntax *object*.**GetCount**

Return type Short Integer

GetLastError S_CON_DOC, S_MAIN_WND

GetCurrentID Method

Description Returns the Unique ID of the current group in the lookup. Returns NULL if no current group exists or on failure.

Object GroupView object

Syntax *object*.**GetCurrentID**

Return type String/BSTR

GetLastError S_CON_DOC, S_GRP_FRM

Example

```
'This example returns the Unique ID of the current group.
Dim objApp as object
Dim objViews as object
Dim objGroup as object
Dim i as integer

'Initialize the Application object.
'This starts ACT! if it is not already running.
Set objApp = CreateObject("ACTOLE.APPOBJECT")

'Open the database.
objApp.OpenFile C:\My Documents\ACT\Database\ACT5demo.dbf

'Create a Views object.
Set objViews = App.Views

'Create the GroupView object. This brings up the Groups view.
Set objGroup = objViews.Create(3, "MyGroup")

objGrp.MoveFirst'Move to the first Group.
```

```
'Go through all the groups and list their Unique IDs and the names of
'the groups.
'Get the number of groups.
For i = 1 To objGrp.GetCount
    'List the group Unique ID and group name.
    List1.AddItem objGrp.GetCurrentID & " : " & objGrp.GetField(GF_Name)
    objGrp.MoveNext
Next i

'Close the Groups view.
objGroup.Close
Set objGroup = Nothing
Set objViews = Nothing
'Close the Application object.
Set objApp = Nothing
```

GetField Method

Description Returns the value of the specified field in the current group record. Returns NULL on failure.

Object GroupView object

Syntax *object*.**GetField** (*lFieldID*)

Parameters *lFieldID* A long integer representing the field ID of the field whose value is to be retrieved. See ACT! Databases for lists of field IDs and names (Field constants).

Return type String/BSTR

GetLastError S_CON_DOC, S_MAIN_WND, S_ACTIVE_REC, S_GETFIELD_FAIL

See also SetField Method

GetSubGroupCount Method

Requires ACT! 5.0 or later

Description Returns the number of subgroups for the current parent group.

Object GroupView object

Syntax *object*.**GetSubGroupCount**

Return type Long Integer

See also AddNewSubGroup Method, ChangeToParentGroup Method,ChangeToSubGroup Method, Collapse Method, Expand Method, GetSubGroupCount Method, GroupType Method, IsExpanded Method

Example

```
'Get the Views object.
Set objViews = objApp.Views

'Create the Groups view.
Set objGrp = objViews.Create(3, "Group")

'Move to the first group.
objGrp.MoveFirst

'Get the number of subgroups in the current group.
count = objGrp.GetSubGroupCount
list1.AddItem "First Group " & objGrp.GetField(GF_Name) & " has "
    & count & "sub groups"
```

GetTabCount Method

Requires ACT! 4.0 or later

Description Returns the number of tabs in the Contact view, including tabs added using the Adding Extensible Views and Tabs to ACT! component of the ACT! SDK. Returns -1 on error.

Object GroupView object

Syntax *object*.**GetTabCount**

Return type Short Integer

GetLastError S_CON_DOC, S_FRM_VW_SYS, S_NO_TAB_CONTAINER

See also GetTabName Method

Example See GetTabCount Method in ContactView object.

GetTabName Method

Requires ACT! 4.0 or later

Description Returns a string containing the name of the specified tab in the Contact view. Returns NULL on error.

Object GroupView object

Syntax *object*.**GetTabName** (*iTabNo*)

Parameters *iTabNo* A short integer that specifies the tab index number, in a range between 0 and one less than the value returned by GetTabCount.

Return type String/BSTR

GetLastError S_CON_DOC, S_FRM_VW_SYS, S_NO_TAB_CONTAINER

See also GetTabCount Method

Example See GetTabCount Method in ContactView object.

Goto Method

Description Displays the group specified by the group Unique ID. Returns S_ERROR on failure.

Object GroupView object

Syntax *object*.**Goto** (*szGroupID*)

Parameters *szGroupID* A string representing the group Unique ID.

Return type Long Integer

GetLastError S_JUMP_TO_REC

See also MoveFirst Method, MoveLast Method, MoveNext Method, MovePrevious Method

GroupType Method

Requires ACT! 5.0 or later

Description Returns 0 if the current group is a parent group or 1 if it is a subgroup.

Object GroupView object

Syntax *object*.**GroupType**

Return type Short Integer

See also AddNewSubGroup Method, ChangeToParentGroup Method,ChangeToSubGroup Method, Collapse Method, Expand Method, GetSubGroupCount Method, IsExpanded Method

Example

```
Set objViews = objApp.Views
Set objGrp = objViews.Create(3, "Group")

objGrp.MoveFirst
While objGrp.EOL <> True
    If objGrp.GroupType = 0 Then
        list1.AddItem objGrp.GetField(GF_Name) & " is a parent group"
    Else
        list1.AddItem objGrp.GetField(GF_Name) & " is a sub group"
    End If
objGrp.MoveNext
Wend
```

IsExpanded Method

Requires ACT! 5.0 or later

Description Returns True if the current group node of the tree is expanded or False if it is collapsed.

Object GroupView object

Syntax *object*.**IsExpanded**

Return type Boolean

See also AddNewSubGroup Method, ChangeToParentGroup Method,ChangeToSubGroup Method, Collapse Method, Expand Method, GetSubGroupCount Method, GroupType Method

Example

```
Set objViews = objApp.Views
Set objGrp = objViews.Create(3, "Group")

objGrp.MoveFirst
While objGrp.EOL <> True
    If objGrp.IsExpanded = False Then
        'Tree for that particular group is collapsed.
        objGrp.Expand
    Else
        objGrp.Collapse
    End If
objGrp.MoveNext
Wend
```

LookupAll Method

Description Creates and displays a lookup of all group records. The result of this lookup becomes the current lookup; the previous lookup is saved. Returns S_ERROR on failure.

Object GroupView object

Syntax *object*.**LookupAll**

Return type Long Integer

GetLastError S_MAIN_WND

See also LookupFieldEx Method, LookupPrevious Method, RunQuery Method

LookupMyRecord Method in the ContactView object

LookupFieldEx Method

Requires ACT! 4.0 or later

Description Creates and displays a group lookup based on a specified field value. The result becomes the current lookup and the previous lookup is saved. Returns S_ERROR on failure.

Note: This method extends the functionality of LookupField in ACT! 4.0 or later.

Object GroupView object

Syntax *object.***LookupFieldEx** (*lFieldID, szFieldValue, iType)*

Parameters *lFieldID* A long integer representing the field ID of the field to be used in the lookup. See ACT! Databases for lists of field IDs and names (Field constants).

szFieldValue A string representing the field value to match. Only groups with the specified field value in the specified field are retrieved.

iType A short integer indicating the action on the current lookup.

The following table lists the values for this parameter.

Value	Setting
0	Add to the current lookup
1	Narrow the current lookup
n	Specify any other number to replace the current lookup

Return type Long Integer

GetLastError S_CON_DOC, S_INVALID_ID

See also LookupAll Method, LookupPrevious Method, RunQuery Method

LookupMyRecord Method in the ContactView object

LookupPrevious Method

Description Returns the most recent lookup. Replaces the current lookup with the most recent lookup. Returns S_ERROR on failure.

Object GroupView object

Syntax *object.***LookupPrevious**

Return type Long Integer

GetLastError S_MAIN_WND

See also LookupAll Method, LookupFieldEx Method, RunQuery Method

LookupMyRecord Method in the ContactView object

MoveFirst Method

Description Displays the first group record in the current lookup. Returns S_ERROR on failure.

Object GroupView object

Syntax *object.***MoveFirst**

Return type Long Integer

GetLastError S_MAIN_WND

See also BOL Method, EOL Method, MoveLast Method, MoveNext Method, MovePrevious Method

MoveLast Method

Description Displays the last group record in the current lookup. Returns S_ERROR on failure.

Object GroupView object

Syntax *object*.**MoveLast**

Return type Long Integer

GetLastError S_MAIN_WND

See also BOL Method, EOL Method, MoveFirst Method, MoveNext Method, MovePrevious Method

MoveNext Method

Description Displays the next group record in the current lookup. Returns S_NONE if there is no next record and S_ERROR on failure.

Object GroupView object

Syntax *object*.**MoveNext**

Return type Long Integer

GetLastError S_MAIN_WND

See also BOL Method, EOL Method, MoveFirst Method, MoveLast Method, MovePrevious Method

Example

```
'This example displays the next group record.
objGroup.MoveNext
If objGroup.EOL then
   MsgBox "End of Group List"
End If
```

MovePrevious Method

Description Displays the previous group record in the current lookup. Returns S_NONE if there is no previous record and S_ERROR on failure.

Object GroupView object

Syntax *object*.**MovePrevious**

Return type Long Integer

GetLastError S_MAIN_WND

See also BOL Method, EOL Method, MoveFirst Method, MoveLast Method, MoveNext Method

NotesHistory Method

Description Returns a dispatch pointer to a grid object for the Notes/History tab. The Notes/History tab must be the active tab before calling this method.

Object GroupView object

Syntax *object*.**NotesHistory**

Return type Object/LPDISPATCH

GetLastError S_NO_DALLIST_VIEW, S_MEM_ERROR, S_CON_DOC, S_FRM_VW_SYS

See also Activities Method, GetActiveTab Method, SetActiveTab Method

RunQuery Method

Description Loads and runs the specified saved query and creates and displays a lookup based on the query. Returns S_ERROR on failure.

Object GroupView object

Syntax *object*.**RunQuery** (*szQueryFile*)

Parameters *szQueryFile* A string representing the complete path and file name of the file containing the query to be executed.

Return type Long Integer

GetLastError S_CON_DOC, S_MAIN_WND

See also SaveQuery Method

Example

```
'This example loads and runs MYQUERY.QRY.
Dim objApp as object
Dim objViews as object
Dim objGroup as object
Dim i as long

'Initialize the Application object.
'This starts ACT! if it is not already running.
Set objApp = CreateObject("ACTOLE.APPOBJECT")

'Open the database.
objApp.OpenFile C:\My Documents\ACT\Database\ACT5demo.dbf

'Create a Views object.
Set objViews = App.Views

'Create the GroupView object. This brings up the Group view.
Set objGroup = objViews.Create(3, "MyGroup")

'Load and run MYQUERY. The group that meets the criteria is displayed.
objGrp.RunQuery "C:\PROGRAM FILES\ACT\QUERY\MYQUERY.QRY"

'Close the Group view.
objGroup.Close
objViews.CloseAll
Set objGroup = Nothing
Set objViews = Nothing
'Close the Application object.
Set objApp = Nothing
```

SaveQuery Method

Description Saves the current query in the specified .QRY file. Returns S_ERROR on failure.

Object GroupView object

Syntax *object*.**SaveQuery** (*szQueryFile*)

Parameters szQueryFile A string representing the complete path and file name of the file where the query will be saved. Query files must have a .QRY extension.

Return type Long Integer

GetLastError S_CON_DOC, S_ACTIVE_REC

See also RunQuery Method

SetActiveTab Method

Description Sets the specified tab as the active tab.

Object GroupView object

Syntax *object*.**SetActiveTab** (*szTabName*)

Parameters szTabName A string representing the name of the tab to be made active.

Return type Long Integer

GetLastError S_INVALID_TAB, S_NO_TAB_CONTAINER, S_CON_DOC

See also Activities Method, GetActiveTab Method, NotesHistory Method

Example See GetActiveTab.

SetField Method

Description Sets the value of the specified field in the current group record.

Object GroupView object

Syntax *object*.**SetField** (*lFieldID*, *szFieldValue*)

Parameters *lFieldID* A long integer representing the field ID of the field whose value is to be set. See ACT! Databases for lists of field IDs and names (Field constants).

szFieldValue A string representing the value to set in the specified field.

Return type Long Integer

GetLastError S_CON_DOC, S_MAIN_WND, S_ACTIVE_REC, S_SETFIELD_FAIL

See also GetField Method

Example

```
'This example goes to a specific group by its group ID and
'sets the address fields.

Dim objApp as object
Dim objViews as object
Dim objGroup as object

'Initialize the Application object.
'This starts ACT! if it is not already running.
Set objApp = CreateObject("ACTOLE.APPOBJECT")

'Open the database.
objApp.OpenFile C:\My Documents\ACT\Database\ACT5demo.dbf

'Create a Views object.
Set objViews = App.Views

'Create the GroupView object. This brings up the Group view.
Set objGroup = objViews.Create(3, "MyGroup")

'Go to the Group with group Unique ID of GUID.
objGrp.Goto GUID

'Set the Address, City, State, and Zip fields.
objGrp.SetField GF_Address1, "20300 Stevens Creek Blvd"
objGrp.SetField GF_City, "Cupertino"
objGrp.SetField GF_State, "CA"
objGrp.SetField GF_Zip, "95014"
```

```
'Close the Group view.
objGroup.Close
objViews.CloseAll
Set objGroup = Nothing
Set objViews = Nothing
'Close the Application object.
Set objApp = Nothing
```

Preferences object

The Preferences object provides an interface to manipulate user preferences in the ACT! application. The following properties and methods apply only to the Preferences object.

Note The Preferences object cannot be used by users with Browse security level. Users must have Standard or Administrator security level to use the Preferences object.

Properties

Name	Parameter(s)	Parameter type(s)	Value type	Access
AttachMsgToContact Property	[= True|False]	Boolean	Boolean	Read/Write
AttachToMsgUsing Property	[= iFormat]	Short Integer	Short Integer	Read/Write
CalendarStartTime Property	[= szTime]	String/BSTR	String/BSTR	Read/Write
CalendarWeekStartsOn Property	[= szDay]	String/BSTR	String/BSTR	Read/Write
CalMinDurationForBanner Property	[= lDuration]	Long Integer	Long Integer	Read/Write
CheckScheduleConflicts Property	[= True|False]	Boolean	Boolean	Read/Write
ContactSalutation Property	[= lSalutation]	Long Integer	Long Integer	Read/Write
DefaultContactLayout Property	[= szLayout]	String/BSTR	String/BSTR	Read/Write
DefaultGroupLayout Property	[= szLayout]	String/BSTR	String/BSTR	Read/Write
DisplayCountryCode Property	[= True|False]	Boolean	Boolean	Read/Write
EnableSpeedLoader Property	[= True|False]	Boolean	Boolean	Read/Write
ExitPrompt Property	[= True|False]	Boolean	Boolean	Read/Write
GenerateSynchReport Property	[= True|False]	Boolean	Boolean	Read/Write
NewActivitiesPrivate Property	[= True|False]	Boolean	Boolean	Read/Write
NewActivitiesSeparate Property	[= True|False]	Boolean	Boolean	Read/Write
NewContactsPrivate Property	[= True|False]	Boolean	Boolean	Read/Write
NewGroupsPrivate Property	[= True|False]	Boolean	Boolean	Read/Write
PromptToPrintEnvelope Property	[= True|False]	Boolean	Boolean	Read/Write
ReceivedSynchLocation Property	[= szLocation]	String/BSTR	String/BSTR	Read/Write
RememberPassword Property	[= True|False]	Boolean	Boolean	Read/Write
RemindToBackup Property	[= lDays]	Long Integer	Long Integer	Read/Write
ReturnReceipt Property	[= True|False]	Boolean	Boolean	Read/Write
SecondGroupColumn Property	[= szClName]	String/BSTR	String/BSTR	Read/Write
ShowContactParsingDialog Property	[= True|False]	Boolean	Long Integer	Read/Write

Name	Parameter(s)	Parameter type(s)	Value type	Access
ShowCurrentMonthOnly Property	[= True|False]	Boolean	Boolean	Read/Write
StartupDatabase Property	[= szDBName]	String/BSTR	String/BSTR	Read/Write
StartupMacro Property	[= szMacroName]	String/BSTR	String/BSTR	Read/Write
TabNavigation Property	[= True|False]	Boolean	Boolean	Read/Write
UseAct20Keys Property	[= True|False]	Boolean	Boolean	Read/Write
UseLastDBonStartup Property	[= True|False]	Boolean	Boolean	Read/Write
UseTypeahead Property	[= True|False]	Boolean	Boolean	Read/Write
WaitTime Property	[= lMinutes]	Long Integer	Long Integer	Read/Write

AttachMsgToContact Property

Requires ACT! 4.0 to ACT! 5.04

Description Gets and sets the mode for attaching new e-mail messages to contacts. Use this property to get, select, or deselect (that is, enable or disable) the Attach To Contact(s) option under New Message Settings in the E-mail tab of the Preferences dialog box. Returns True if the AttachTo Contact(s) option is selected or False if this option is deselected. Returns False on failure.

Object Preferences object

Syntax *object*.**AttachMsgToContact** [= *True*|*False*]

Parameters *True*|*False* Specify True to select the Attach To Contact(s) option or False to deselect this option. Omit this optional parameter to get the setting (selected or deselected) of the Attach To Contact(s) option.

Value type Boolean, Read/Write

GetLastError S_NO_SECURITY, S_NO_USER, S_NOT_PRIVILEGED, S_PROPS_NOT_FOUND

Example

```
Dim objApp As Object
Dim objPref As Object

'Open the database with the default database.
Set objApp = CreateObject("ACTOLE.APPOBJECT")
'Open a database.
objApp.OpenDB C:\My Documents\ACT\Database\ACT5demo.dbf

'Get the Preferences object from the Application object.
Set objPref = objApp.Preferences
'Set the property AttachMsgToContact to True.
objPref.AttachMsgToContact True

Set objPref = Nothing
Set objApp = Nothing
```

AttachToMsgUsing Property

Requires ACT! 4.0 to ACT! 5.04

Description Gets and sets the mode for the format to use when attaching contacts or activities to e-mail messages. Use this property to get, select, or deselect (that is, enable or disable) the Attaching Contacts/activities To Messages options in the General tab of the Preferences dialog box. Returns S_ERROR on failure.

Object Preferences object

Syntax *object.***AttachToMsgUsing** [= *iFormat*]

Parameters *iFormat* A short integer representing the format to use when attaching contacts or activities to e-mail messages.

The following table lists the values that are set or returned for this property.

Value	Option
1	ACT! data (for use with other ACT! users)
2	vCard/vCalendar (MS Outlook compatible)
3	Attach using both formats

Value type Short Integer, Read/Write

GetLastError S_INVALID_INPUT, S_NO_SECURITY, S_NO_USER, S_NOT_PRIVILEGED, S_PROPS_NOT_FOUND, S_WRITE_FAIL

Example

```
Dim objApp As Object
Dim objPref As Object

'Open the database with the default database.
Set objApp = CreateObject("ACTOLE.APPOBJECT")

'Open a database.
objApp.OpenDB C:\My Documents\ACT\Database\ACT5demo.dbf

'Get the Preferences object from the Application object.
Set objPref = objApp.Preferences

'Set the property AttachToMsgUsing to 1 (use ACT! data format).
objPref.AttachToMsgUsing 1

Set objPref = Nothing
Set objApp = Nothing
```

CalendarStartTime Property

Description Gets and sets the time of day to begin the daily and weekly calendars. Returns NULL on error.

Object Preferences object

Syntax *object.***CalendarStartTime** [= *szTime]*

Parameters *szTime* A string that specifies the time of day to start the daily and weekly calendars. Omit this optional parameter to get the starting time for the calendars. The format of szTime is "hh:mm AM|PM".

Value type String/BSTR, Read/Write

See also CalendarWeekStartsOn Property

GetLastError S_INVALID_INPUT, S_NO_SECURITY, S_NO_USER, S_NOT_PRIVILEGED, S_PROPS_NOT_FOUND, S_WRITE_FAIL

Example

```
'This example sets the Start Time in the Preferences
'dialog box and accordingly in the ACT! application.
Dim objApp as object
Dim objPreferences as object
```

```
'Initialize the Application object.
'This starts ACT! if it is not already running.
Set objApp = CreateObject("ACTOLE.APPOBJECT")

'Open the database.
objApp.OpenFile C:\My Documents\ACT\Database\ACT5demo.dbf

'Set the Preferences object.
Set objPreferences = objApp.Preferences

'Set the Calendar Start Time to 7:00AM (Calendars tab).
objPreferences.CalendarStartTime = "7:00AM"

'Close the application.
Set objPreferences = Nothing
Set objApp = Nothing
```

CalendarWeekStartsOn Property

Description Gets and sets the starting day of the week of the calendars. Returns NULL on error.

Object Preferences object

Syntax *object*.**CalendarWeekStartsOn** [= *szDay*]

Parameters *szDay* A string that specifies the starting day of the week. Omit this optional parameter to get the starting day. The parameter szTime can be "Sunday" or "Monday".

Value type String/BSTR, Read/Write

See also CalendarStartTime Property

GetLastError S_NO_SECURITY, S_NO_USER, S_NOT_PRIVILEGED, S_PROPS_NOT_FOUND, S_WRITE_FAIL

Example

```
'This example sets the starting day for the monthly calendar in the
'Preferences dialog box and accordingly in the ACT! application.
Dim objApp as object
Dim objPreferences as object

'Initialize the Application object.
'This starts ACT! if it is not already running.
Set objApp = CreateObject("ACTOLE.APPOBJECT")

'Open the database.
objApp.OpenFile C:\My Documents\ACT\Database\ACT5demo.dbf

'Set the Preferences object.
Set objPreferences = objApp.Preferences

'Select the Sunday option in the Calendar Week Starts On group
'box (Calendars tab).
objPreferences.CalendarWeekStartsOn "Sunday"

'Close the application.
Set objPreferences = Nothing
Set objApp = Nothing
```

CalMinDurationForBanner Property

Requires ACT! 4.0 or later

Description Gets and sets a value that represents the minimum duration of activities for which the full day banner is to be displayed. Use this property to get and set the Show Full Day Banner For Activities With Duration Of x Or Longer option in the Preferences Calendars tab, where x is the number of minutes, hours, or days. Returns -1 on failure.

Object Preferences object

Syntax *object*.**CalMinDurationForBanner** [= *lDuration*]

Parameters *lDuration* A long integer in that specifies the minimum duration in minutes for full day banners in the calendar. Omit this optional parameter to get the setting of the Show Full Day Banner For Activities With Duration Of x Or Longer option in the Preferences Calendars tab.

The following table lists the values that are set or returned for this property. If an invalid value is specified, the default setting of 8 hours is used.

Value	Setting	Value	Setting
0	0 minutes	120	2 hours
5	5 minutes	180	3 hours
10	10 minutes	480	8 hours
15	15 minutes	1440	1 day
30	30 minutes	7200	5 days
45	45 minutes	43200	30 days
60	1 hour		

Value type Long Integer, Read/Write

GetLastError S_NO_SECURITY, S_NO_USER, S_NOT_PRIVILEGED, S_PROPS_NOT_FOUND

Example

```
'This example sets 2 hours as the minimum duration of activities
'for which the full day banner is to be displayed.
Dim objApp as object
Dim objPreferences as object

'Initialize the Application object.
'This starts ACT! if it is not already running.
Set objApp = CreateObject("ACTOLE.APPOBJECT")

'Open the database.
objApp.OpenFile C:\My Documents\ACT\Database\ACT5demo.dbf

'Set the Preferences object.
Set objPreferences = objApp.Preferences

'Show full day banner for activities with a duration of 2 hours is set.
objPreferences.CalMinDurationForBanner = 120

'Close the application.
Set objPreferences = Nothing
Set objApp = Nothing
```

CheckScheduleConflicts Property

Description Gets and sets the mode of schedule conflict checking. Use this property to get, select, or deselect (that is, enable or disable) the Enable Activity Conflict Checking option in the Scheduling tab. If this option is selected, an alert message is displayed if an activity is scheduled that conflicts with or overlaps another activity.

Object Preferences object

Syntax *object*.**CheckScheduleConflicts** [= *True*|*False*]

Parameters *True*|*False* Specify True to set schedule conflict checking or False to disable schedule conflict checking. Omit this optional parameter to get the setting (selected or deselected) of the Enable Activity Conflict Checking option.

Value type Boolean, Read/Write

GetLastError S_NO_SECURITY, S_NO_USER, S_NOT_PRIVILEGED, S_PROPS_NOT_FOUND, S_WRITE_FAIL

Example

```
'This example enables activity conflict checking in the Preferences
'dialog box and accordingly in the ACT! application.

Dim objApp as object
Dim objPreferences as object

'Initialize the Application object.
'This starts ACT! if it is not already running.
Set objApp = CreateObject("ACTOLE.APPOBJECT")
'Open the database.
objApp.OpenFile C:\My Documents\ACT\Database\ACT5demo.dbf

'Set the Preferences object.
Set objPreferences = objApp.Preferences

'Select the Enable Activity Conflict Checking option (Scheduling tab).
objPreferences.CheckScheduleConflicts = True

'Close the application.
Set objPreferences = Nothing
Set objApp = Nothing
```

ContactSalutation Property

Requires ACT! 4.0 or later

Description Gets and sets the value that represents the Salutation option to be used with contacts. Use this property to get and set the Salutation option in the Preferences Names tab.

Object Preferences object

Syntax *object*.**ContactSalutation** [= *lSalutation*]

Parameters *lSalutation* A long integer in the range from 0 to 2 that specifies the option for the Salutation type. Omit this optional parameter to get the setting of the Salutation option in the Preferences Names tab. The following table lists the values that are set or returned for this property.

Value	Setting
0	Use Contact's Last Name
1	Use Contact's First Name
2	Do Not Fill Salutation

Value type Long Integer, Read/Write

GetLastError S_INVALID_INPUT, S_NO_SECURITY, S_NO_USER, S_NOT_PRIVILEGED, S_PROPS_NOT_FOUND, S_WRITE_FAIL

Example

```
'This example selects the Use contact's first name option for
'Salutation in Names preferences.

Dim objApp as object
Dim objPreferences as object

'Initialize the Application object.
'This starts ACT! if it is not already running.
Set objApp = CreateObject("ACTOLE.APPOBJECT")

'Open the database.
objApp.OpenFile C:\My Documents\ACT\Database\ACT5demo.dbf

'Set the Preferences object.
Set objPreferences = objApp.Preferences

'Selects the Contacts First Name Option Button.
objPreferences.ContactSalutation 1

'Close the application.
Set objPreferences = Nothing
Set objApp = Nothing
```

DefaultContactLayout Property

Description Gets and sets the name (and path) of the layout file to use for contacts. The file name extension must be .CLY. Returns NULL on error.

Object Preferences object

Syntax *object*.**DefaultContactLayout** [= *szLayout*]

Parameters *szLayout* A string that specifies the name and path of the layout file to be used for contacts. Omit this optional parameter to get the name of the layout file for contacts.

Value type String/BSTR, Read/Write

GetLastError S_NO_SECURITY, S_NO_USER, S_NOT_FOUND, S_NOT_PRIVILEGED, S_PROPS_NOT_FOUND, S_WRITE_FAIL

See also DefaultGroupLayout Property

Example

```
'This example specifies the layout file to use for contacts in the
'Preferences dialog box and accordingly in the ACT! application.

Dim objApp as object
Dim objPreferences as object

'Initialize the Application object.
'This starts ACT! if it is not already running.
Set objApp = CreateObject("ACTOLE.APPOBJECT")

'Open the database.
objApp.OpenFile C:\My Documents\ACT\Database\ACT5demo.dbf
```

```
'Set the Preferences object.
Set objPreferences = objApp.Preferences

'Select C:\PROGRAM FILES\ACT\LAYOUT\CONTACT1.CLY
'in the Default Contact Layout field (Startup tab).
objPreferences.DefaultContactLayout =
    "C:\PROGRAM FILES\ACT\LAYOUT\CONTACT1.CLY"

'Close the application.
Set objPreferences = Nothing
Set objApp = Nothing
```

DefaultGroupLayout Property

Description Gets and sets the name (and path) of the layout file to use for groups. The file name extension must be .GLY. Returns NULL on error.

Object Preferences object

Syntax *object*.**DefaultGroupLayout** [= *szLayout*]

Parameters *szLayout* A string that specifies the name and path of the layout file to be used for groups. Omit this optional parameter to get the name of the layout file for groups.

Value type String/BSTR, Read/Write

GetLastError S_NO_SECURITY, S_NO_USER, S_NOT_FOUND, S_NOT_PRIVILEGED, S_PROPS_NOT_FOUND, S_WRITE_FAIL

See also DefaultContactLayout Property

Example

```
'This example specifies the layout file to use for groups in the
'Preferences dialog box and accordingly in the ACT! application.

Dim objApp as object
Dim objPreferences as object

'Initialize the Application object.
'This starts ACT! if it is not already running.
Set objApp = CreateObject("ACTOLE.APPOBJECT")

'Open the database.
objApp.OpenFile C:\My Documents\ACT\Database\ACT5demo.dbf

'Set the Preferences object.
Set objPreferences = objApp.Preferences

'Select C:\PROGRAM FILES\ACT\LAYOUT\ACCOUNT5.GLY
'in the Default Group Layout list (Startup tab).
objPreferences.DefaultGroupLayout =
    "C:\PROGRAM FILES\ACT\LAYOUT\ACCOUNT5.GLY"

'Close the application.
Set objPreferences = Nothing
Set objApp = Nothing
```

DisplayCountryCode Property

Description Gets and sets the mode for preceding phone numbers with a country code. Use this property to get, select, or deselect (that is, enable or disable) the Always Display Country Code in Phone Fields option in the General tab. If this option is selected, country codes are always displayed before a phone number in phone fields. Returns False on failure.

Object Preferences object

Syntax *object*.**DisplayCountryCode** [= *True*|*False*]

Parameters *True*|*False* Specify True to display a country code before phone numbers or False if phone numbers are not preceded by a country code. Omit this optional parameter to get the setting (selected or deselected) of the Always Display Country Code in Phone Fields option.

Value type Boolean, Read/Write

GetLastError S_NO_SECURITY, S_NO_USER, S_NOT_PRIVILEGED, S_PROPS_NOT_FOUND, S_WRITE_FAIL

Example

```
'This example specifies that phone numbers are prefixed with an
'country code in the Preferences dialog box and accordingly in
'the ACT! application.

Dim objApp as object
Dim objPreferences as object

'Initialize the Application object.
'This starts ACT! if it is not already running.
Set objApp = CreateObject("ACTOLE.APPOBJECT")

'Open the database.
objApp.OpenFile C:\My Documents\ACT\Database\ACT5demo.dbf

'Set the Preferences object.
Set objPreferences = objApp.Preferences

'Set the Always Display Country Code option (General tab).
objPreferences.DisplayCountryCode True

'Close the application.
Set objPreferences = Nothing
Set objApp = Nothing
```

EnableSpeedLoader Property

Requires ACT! 4.0 only

Description Gets and sets the value of the Enable ACT! Speed Loader option in the Startup tab of the Preferences dialog box. Returns True if the Enable ACT! Speed Loader option is selected or False if it is deselected. Returns False on error.

Object Preferences object

Syntax *object*.**EnableSpeedLoader** [= *True*|*False*]

Parameters *True*|*False* Specify True to select the Enable ACT! Speed Loader option or False to deselect it. Omit this optional parameter to get the setting (selected or deselected) of the Enable ACT! Speed Loader option.

Value type Boolean, Read/Write

GetLastError S_NO_SECURITY, S_NO_USER, S_NOT_PRIVILEGED, S_PROPS_NOT_FOUND

Example

```
Dim objApp As Object
Dim objPref As Object

'Open the database with the default database.
Set objApp = CreateObject("ACTOLE.APPOBJECT")

'Open a database.
objApp.OpenDB C:\My Documents\ACT\Database\ACT5demo.dbf

'Get the Preferences object from the Application object.
Set objPref = objApp.Preferences

'Turn off the Speed Loader option
objPref.EnableSpeedLoader False

'Close the Preferences object.
Set objPref = Nothing
'Close the Application object.
Set objApp = Nothing
```

ExitPrompt Property

Description Gets and sets the mode of prompting before a user exits ACT! Use this property to get, enable, or disable (that is, select or deselect) the Prompt Before Exiting option in the General tab. If this option is selected, a confirmation message is displayed when the user exits ACT! Returns False on error.

Object Preferences object

Syntax *object*.**ExitPrompt** [= *True*|*False*]

Parameters *True*|*False* Specify True to prompt the user before exiting ACT! or False to omit the prompt before a user exits ACT! Omit this optional parameter to get the setting (selected or deselected) of the Prompt Before Exiting option.

Value type Boolean, Read/Write

GetLastError S_NO_SECURITY, S_NO_USER, S_NOT_PRIVILEGED, S_PROPS_NOT_FOUND, S_WRITE_FAIL

Example

```
'This example specifies that the user is prompted before quitting ACT!
'in the Preferences dialog box and accordingly in the ACT! application.

Dim objApp as object
Dim objPreferences as object

'Initialize the Application object.
'This starts ACT! if it is not already running.
Set objApp = CreateObject("ACTOLE.APPOBJECT")

'Open the database.
objApp.OpenFile C:\My Documents\ACT\Database\ACT5demo.dbf

'Set the Preferences object.
Set objPreferences = objApp.Preferences

'Select the Prompt Before Exiting option (General tab).
objPreferences.ExitPrompt = True
```

```
'Close the application.
Set objPreferences = Nothing
Set objApp = Nothing
```

GenerateSynchReport Property

Requires ACT! 4.0 or later

Description Gets and sets the Generate Synchronization Report option in the Preferences Synchronization tab. Returns S_ERROR on error.

Object Preferences object

Syntax *object*.**GenerateSynchReport** [= *True*|*False*]

Parameters *True*|*False* Specifies whether a synchronization report is to be generated. Specify True to select the Generate Synchronization Report option or False to deselect it. Omit this optional parameter to get the setting (selected or deselected) of the Generate Synchronization Report option.

Value type Boolean, Read/Write

GetLastError S_NO_SECURITY, S_NO_USER, S_NOT_PRIVILEGED, S_PROPS_NOT_FOUND, S_WRITE_FAIL

NewActivitiesPrivate Property

Description Gets and sets the public/private preference option for new activities. Use this property to get, select, or deselect the Make New Activities Public option in the Scheduling tab. Returns False on error.

Object Preferences object

Syntax *object*.**NewActivitiesPrivate** [= *True*|*False*]

Parameters *True*|*False* Specify True to make all new activities default to private or False to make them default to public. Omit this optional parameter to get the setting (selected or deselected) of the Make New Activities Public option.

Value type Boolean, Read/Write

GetLastError S_NO_SECURITY, S_NO_USER, S_NOT_PRIVILEGED, S_PROPS_NOT_FOUND, S_WRITE_FAIL

Example

```
'This example sets new activities to Public in the Preferences
'dialog box and accordingly in the ACT! application.

Dim objApp as object
Dim objPreferences as object

'Initialize the Application object.
'This starts ACT! if it is not already running.
Set objApp = CreateObject("ACTOLE.APPOBJECT")
'Open the database.
objApp.OpenFile C:\My Documents\ACT\Database\ACT5demo.dbf

'Set the Preferences object.
Set objPreferences = objApp.Preferences

'Select the Default Activities to Public option (Scheduling tab).
objPreferences.NewActivitiesPrivate = False

'Close the application.
Set objPreferences = Nothing
Set objApp = Nothing
```

NewActivitiesSeparate Property

Requires ACT! 4.0 or later

Description Gets and sets the preference option that specifies if a separate activity should be generated for each contact when scheduling new activities with multiple contacts. Use this property to get, select, or deselect the When Scheduling With Multiple Contacts, Always Create Separate Activities For Each option in the Scheduling tab. If this option is selected, by default a separate activity is created for each contact when a new activity is scheduled for multiple contacts.

Object Preferences object

Syntax *object*.**NewActivitiesSeparate** [= *True*|*False*]

Parameters *True*|*False* Specify True to create a separate activity for each contact by default when an activity is scheduled for multiple contacts. Specify False to create a single activity by default for all selected contacts. Omit this optional parameter to get the setting (selected or deselected) of the When Scheduling With Multiple Contacts, Always Create Separate Activities For Each option.

Value type Boolean, Read/Write

GetLastError S_NO_SECURITY, S_NO_USER, S_NOT_PRIVILEGED, S_PROPS_NOT_FOUND, S_WRITE_FAIL

Example

```
'This example sets the When scheduling with multiple contacts,
'always create separate activities for each option.

Dim objApp as object
Dim objPreferences as object

'Initialize the Application object.
'This starts ACT! if it is not already running.
Set objApp = CreateObject("ACTOLE.APPOBJECT")

'Open the database.
objApp.OpenFile C:\My Documents\ACT\Database\ACT5demo.dbf

'Set the Preferences object.
Set objPreferences = objApp.Preferences

'Deselects the When scheduling with multiple contacts, always create
'separate activities for each option in the Scheduling Tab.
objPreferences.NewActivitiesSeparate False

'Close the application.
Set objPreferences = Nothing
Set objApp = Nothing
```

NewContactsPrivate Property

Description Gets and sets the public/private mode for new contacts. Use this property to get, select, or deselect the Make New Contacts Private option in the Startup tab. If this option is selected, all new activities default to public activities.

Object Preferences object

Syntax *object*.**NewContactsPrivate** [= *True*|*False*]

Parameters *True*|*False* Specify True to make all new contacts private or False to make them public. Omit this optional parameter to get the setting (selected or deselected) of the Make New Contacts Private option.

Value type Boolean, Read/Write

GetLastError S_NO_SECURITY, S_NO_USER, S_NOT_PRIVILEGED, S_PROPS_NOT_FOUND, S_WRITE_FAIL

Example

```
'This example makes new contacts Public in the Preferences
'dialog box and accordingly in the ACT! application.

Dim objApp as object
Dim objPreferences as object

'Initialize the Application object.
'This starts ACT! if it is not already running.
Set objApp = CreateObject("ACTOLE.APPOBJECT")

'Open the database.
objApp.OpenFile C:\My Documents\ACT\Database\ACT5demo.dbf

'Set the Preferences object.
Set objPreferences = objApp.Preferences

'Deselect the Make New Contacts Private option (Startup tab).
objPreferences.NewContactsPrivate = False

'Close the application.
Set objPreferences = Nothing
Set objApp = Nothing
```

NewGroupsPrivate Property

Description Gets and sets the public/private mode for new groups. Use this property to get, enable, or disable (that is, select or deselect) the Make New Groups Private option in the Startup tab. If this option is selected, all new activities are public activities.

Object Preferences object

Syntax *object*.**NewGroupsPrivate** [= *True*|*False*]

Parameters *True*|*False* Specify True to make all new groups private or False to make them public. Omit this optional parameter to get the setting (selected or deselected) of the Make New Groups Private option.

Value type Boolean, Read/Write

GetLastError S_NO_SECURITY, S_NO_USER, S_NOT_PRIVILEGED, S_PROPS_NOT_FOUND, S_WRITE_FAIL

Example

```
'This example sets new groups to Public in the Preferences
'dialog box and accordingly in the ACT! application.

Dim objApp as object
Dim objPreferences as object

'Initialize the Application object.
'This starts ACT! if it is not already running.
Set objApp = CreateObject("ACTOLE.APPOBJECT")

'Open the database.
objApp.OpenFile C:\My Documents\ACT\Database\ACT5demo.dbf
```

```
'Set the Preferences object.
Set objPreferences = objApp.Preferences

'Deselect the Make New Groups Private (Startup tab)
objPreferences.NewGroupsPrivate = False

'Close the application.
Set objPreferences = Nothing
Set objApp = Nothing
```

PromptToPrintEnvelope Property

Requires ACT! 4.0 or later

Description Gets and sets the mode for the prompt for printing an envelope when printing a letter. Use this property to get, enable, or disable (that is, select or deselect) the When Printing Letters, Prompt To Print An Envelope option in the General tab of the Preferences dialog box. Returns True if the When Printing Letters, PromptTo Print An Envelope option is selected or False if this option is deselected. Returns False on failure.

Object Preferences object

Syntax *object*.**PromptToPrintEnvelope** [= *True*|*False*]

Parameters *True*|*False* Specify True to select the When Printing Letters, PromptTo Print An Envelope option or False to deselect this option. Omit this optional parameter to get the setting (selected or deselected) of the When Printing Letters, Prompt To Print An Envelope option.

Value type Boolean, Read/Write

GetLastError S_NO_SECURITY, S_NO_USER, S_NOT_PRIVILEGED, S_PROPS_NOT_FOUND

Example

```
Dim objApp As Object
Dim objPref As Object
Dim strSynchScheduleInfo

'Open the database with the default database.
Set objApp = CreateObject("ACTOLE.APPOBJECT")

'Open a database.
objApp.OpenDB C:\My Documents\ACT\Database\ACT5demo.dbf

'Get the Preferences object from the Application object.
Set objPref = objApp.Preferences

'Set the property PromptToPrintEnvelope to True to get the prompt.
objPref.PromptToPrintEnvelope True

'Close the Preferences object.
Set objPref = Nothing
'Close the Application object.
Set objApp = Nothing
```

ReceivedSynchLocation Property

Requires ACT! 4.0 or later

Description Gets and sets the All Received Synchronizations Go option in the Preferences Synchronization tab. Returns NULL on error. The location is verified to ensure it exists before the property is set.

Object Preferences object

Syntax *object*.**ReceivedSynchLocation** [= *szLocation*]

Parameters *szLocation* A string that specifies the new location of the folder for received synchronizations to be stored. Omit this optional parameter to get the current location.

Value type String/BSTR, Read/Write

GetLastError S_NO_SECURITY, S_NO_USER, S_NOT_PRIVILEGED, S_PROPS_NOT_FOUND, S_WRITE_FAIL

RememberPassword Property

Description Gets and sets the mode for remembering passwords. Use this property to get, enable, or disable (that is, select or deselect) the Remember Password option in the General tab. If this option is selected, ACT! does not prompt the user for a password.

Object Preferences object

Syntax *object*.**RememberPassword** [= *True*|*False*]

Parameters *True*|*False* Specify True to remember the password or False to prompt the user for a password. Omit this optional parameter to get the setting (selected or deselected) of the Remember Password option.

Value type Boolean, Read/Write

GetLastError S_NO_SECURITY, S_NO_USER, S_NOT_PRIVILEGED, S_PROPS_NOT_FOUND, S_WRITE_FAIL

Example

```
'This example specifies that ACT! remembers the user password in the
'Preferences dialog box and accordingly in the ACT! application.

Dim objApp as object
Dim objPreferences as object

'Initialize the Application object.
'This starts ACT! if it is not already running.
Set objApp = CreateObject("ACTOLE.APPOBJECT")

'Open the database.
objApp.OpenFile C:\My Documents\ACT\Database\ACT5demo.dbf

'Set the Preferences object.
Set objPreferences = objApp.Preferences

'Set the Remember Password option (General tab).
objPreferences.RememberPassword = True

'Close the application.
Set objPreferences = Nothing
Set objApp = Nothing
```

RemindToBackup Property

Requires ACT! 4.0 only.

Description Gets and sets the value of the Remind Me To Backup Every n Days option in the General tab of the Preferences dialog box. Returns False on failure.

Object Preferences object

Syntax *object*.**RemindToBackup** [= *lDays*]

Parameters *lDays* A long integer representing the number of days for the Remind Me To Backup Every n Days option, in the range from 0 to 365. Specify a value of 0 to deselect this option and disable the reminder. If this parameter is omitted, this property returns the number of days if the Remind Me To Backup Every n Days option is selected or a value of 0 if this option is deselected.

Value type Long Integer, Read/Write

GetLastError S_INVALID_INPUT, S_NO_SECURITY, S_NO_USER, S_NOT_PRIVILEGED, S_PROPS_NOT_FOUND

See also BackupDB Method and RestoreDB Method in the Application object

Example

```
Dim objApp As Object
Dim objPref As Object

'Open the database with the default database.
Set objApp = CreateObject("ACTOLE.APPOBJECT")

'Open a database.
objApp.OpenDB C:\My Documents\ACT\Database\ACT5demo.dbf

'Get the Preferences object from the Application object.
Set objPref = objApp.Preferences

'Set the schedule for backup reminder to 20 days.
objPref.RemindToBackUp 20

'Close the Preferences object.
Set objPref = Nothing
'Close the Application object.
Set objApp = Nothing
```

ReturnReceipt Property

Requires ACT! 4.0 or later

Description Gets and sets the mode for receiving return receipts from recipients of e-mail messages. Use this property to get, enable, or disable (that is, select or deselect) the Return Receipt option under New Message Settings in the E-mail tab of the Preferences dialog box. Returns True if the Return Receipt option is selected or False if this option is deselected. Returns False on failure.

Object Preferences object

Syntax *object*.**ReturnReceipt** [= *True*|*False*]

Parameters *True*|*False* Specify True to select the Return Receipt option or False to deselect it. Omit this optional parameter to get the setting (selected or deselected) of the Return Receipt option.

Value type Boolean, Read/Write

GetLastError S_NO_SECURITY, S_NO_USER, S_NOT_PRIVILEGED, S_PROPS_NOT_FOUND

Example

```
Dim objApp As Object
Dim objPref As Object

'Open the database with the default database.
Set objApp = CreateObject("ACTOLE.APPOBJECT")
'Open a database.
objApp.OpenDB  C:\My Documents\ACT\Database\ACT5demo.dbf
```

```
'Get the Preferences object from the Application object.
Set objPref = objApp.Preferences

'Select the return receipt option.
objPref.ReturnReceipt True

'Close the Preferences object.
Set objPref = Nothing
'Close the Application object.
Set objApp = Nothing
```

SecondGroupColumn Property

Description Gets and sets the name of the second column in the Group view. Returns NULL on error.

Object Preferences object

Syntax *object*.**SecondGroupColumn** [= *szClName*]

Parameters *szClName* A string that specifies the name of the second column to be displayed in the Group view. Omit this optional parameter to get the name of the second column.

Value type String/BSTR, Read/Write

GetLastError S_CON_DOC, S_INVALID_INPUT, S_NO_DB, S_NO_SCHEMA, S_NO_SECURITY, S_NO_USER, S_NOT_PRIVILEGED, S_PROPS_NOT_FOUND, S_WRITE_FAIL

Example

```
'This example sets City as the second column in the Group in the
'Preferences dialog box and accordingly in the ACT! application.

Dim objApp as object
Dim objPreferences as object

'Initialize the Application object.
'This starts ACT! if it is not already running.
Set objApp = CreateObject("ACTOLE.APPOBJECT")

'Open the database.
objApp.OpenFile C:\My Documents\ACT\Database\ACT5demo.dbf

'Set the Preferences object.
Set objPreferences = objApp.Preferences

'Select City in the Second Group Column combo box (Startup tab).
objPreferences.SecondGroupColumn "City"

'Close the application.
Set objPreferences = Nothing
Set objApp = Nothing
```

ShowContactParsingDialog Property

Requires ACT! 5.0 or later

Description Automatically shows the contact name definition dialog if the contact name contains more than two names (Chester Van Houten, for example). Returns 0 if the preference is successfully set or an error code if unsuccessful. For more information, see Error codes.

Object Preferences object

Syntax *object*.**ShowContactParsingDialog** (= *True*|*False*)

Parameters = *True*|*False* Specify True to show the dialog or False to not show it.

Value type Long Integer

Example

```
Dim objApp as object
Dim objPreferences as object

'Initialize the Application object.
'This starts ACT! if it is not already running.
Set objApp = CreateObject("ACTOLE.APPOBJECT")

'Open the database.
objApp.OpenFile C:\My Documents\ACT\Database\ACT5demo.dbf

'Set the Preferences object.
Set objPreferences = objApp.Preferences

'Set the preferences to Show Contact Name Parsing dialog
objPreferences.ShowContactParsingDialog True

'Close the application.
Set objPreferences = Nothing
Set objApp = Nothing
```

ShowCurrentMonthOnly Property

Description Gets and sets the mode for displaying the Mini-Calendar. Use this property to get, enable, or disable (that is, select or deselect) the When Displaying Mini-Calendar, Show Only Current Month option in the Calendars tab. If this option is selected, ACT! displays only one month, instead of three, in the Mini-Calendar.

Object Preferences object

Syntax *object*.**ShowCurrentMonthOnly** [= *True*|*False*]

Parameters *True*|*False* Specify True to display only one month in the Mini-Calendar or False to display three months. Omit this optional parameter to get the setting (selected or deselected) of the When Displaying Mini-Calendar, Show Only Current Month option.

Value type Boolean, Read/Write

GetLastError S_NO_SECURITY, S_NO_USER, S_NOT_PRIVILEGED, S_PROPS_NOT_FOUND, S_WRITE_FAIL

Example

```
'This example specifies that only the current month should be
'displayed in the Mini-Calendar in the Preferences dialog box and
'accordingly in the ACT! application.

Dim objApp as object
Dim objPreferences as object

'Initialize the Application object.
'This starts ACT! if it is not already running.
Set objApp = CreateObject("ACTOLE.APPOBJECT")

'Open the database.
objApp.OpenFile C:\My Documents\ACT\Database\ACT5demo.dbf

'Set the Preferences object.
Set objPreferences = objApp.Preferences
```

```
'Select the When Displaying Mini-Calendar, Show Only Current Month
'option (Calendars tab).
objPreferences.ShowCurrentMonthOnly True

'Close the application.
Set objPreferences = Nothing
Set objApp = Nothing
```

StartupDatabase Property

Description Gets and sets the name of the database that opens when ACT! is started. The database must exist when this property is used. This property is effective only if UseLastDBonStartup is set to False.

Object Preferences object

Syntax *object*.**StartupDatabase** [= *szDBName*]

Parameters *szDBName* A string that specifies the database to be opened when ACT! is started. Omit this optional parameter to get the name of the database.

Value type String/BSTR, Read/Write

GetLastError S_NO_SECURITY, S_NO_USER, S_NOT_FOUND, S_NOT_PRIVILEGED, S_PROPS_NOT_FOUND, S_WRITE_FAIL

See also UseLastDBonStartup Property, StartupMacro Property

Example

```
'This example specifies the startup database in the Preferences
'dialog box and accordingly in the ACT! application.

Dim objApp as object
Dim objPreferences as object

'Initialize the Application object.
'This starts ACT! if it is not already running.
Set objApp = CreateObject("ACTOLE.APPOBJECT")

'Open the database.
objApp.OpenFile C:\My Documents\ACT\Database\ACT5demo.dbf

'Set the Preferences object.
Set objPreferences = objApp.Preferences

'C:\My Documents\ACT\Database\ACT5demo.dbf
'is the startup database (Startup tab).
objPreferences.StartupDatabase = C:\My Documents\ACT\Database\ACT5demo.dbf

'Close the application.
Set objPreferences = Nothing
Set objApp = Nothing
```

StartupMacro Property

Description Gets and sets the full path and file name of a file containing a macro that ACT! executes at startup. The macro file must exist when this property is used.

Object Preferences object

Syntax *object*.**StartupMacro** [= *szMacroName*]

Parameters *szMacroName*A string that specifies the macro file name to be executed when ACT! is started. Omit this optional parameter to get the name of the macro.

GetLastError S_NO_SECURITY, S_NO_USER, S_NOT_FOUND, S_NOT_PRIVILEGED, S_PROPS_NOT_FOUND, S_WRITE_FAIL

Value type String/BSTR, Read/Write

TabNavigation Property

Description Gets and sets the key used to move between fields.

Object Preferences object

Syntax *object*.**TabNavigation** [= *True*|*False*]

Parameters *True*|*False* Specify True to use the Tab key to move between fields or False to use the Enter key. Omit this optional parameter to get the key used to move between fields.

Value type Boolean, Read/Write

GetLastError S_NO_SECURITY, S_NO_USER, S_NOT_PRIVILEGED, S_PROPS_NOT_FOUND, S_WRITE_FAIL

Example

```
'This example specifies that the Tab key is used to move between fields
'in the Preferences dialog box and accordingly in the ACT! application.

Dim objApp as object
Dim objPreferences as object

'Initialize the Application object.
'This starts ACT! if it is not already running.
Set objApp = CreateObject("ACTOLE.APPOBJECT")

'Open the database.
objApp.OpenFile C:\My Documents\ACT\Database\ACT5demo.dbf

'Set the Preferences object.
Set objPreferences = objApp.Preferences

'Select the Tab Key option in the Move Between Fields Using
'group box (General tab).
objPreferences.TabNavigation = True

'Close the application.
Set objPreferences = Nothing
Set objApp = Nothing
```

UseAct20Keys Property

Description Gets and sets the mode of the Move Between Records Using ACT! 2.0 Shortcut Keys option. Use this property to get, select, or deselect (that is, enable or disable) this option in the General tab of the Preferences dialog box.

Object Preferences object

Syntax *object*.**UseAct20Keys** [= *True*|*False*]

Parameters *True*|*False* Specify True to use ACT! 2.0 shortcut keys to move between records or False to use ACT! 5.0 shortcut keys. Omit this optional parameter to get the setting (selected or deselected) of the Move Between Records Using ACT! 2.0 Shortcut Keys option.

Value type Boolean, Read/Write

GetLastError S_NO_SECURITY, S_NO_USER, S_NOT_PRIVILEGED, S_PROPS_NOT_FOUND, S_WRITE_FAIL

Example

```
'This example specifies that ACT!2.0 shortcut keys are used in the
'Preferences dialog box and accordingly in the ACT! application.

Dim objApp as object
Dim objPreferences as object

'Initialize the Application object.
'This starts ACT! if it is not already running.
Set objApp = CreateObject("ACTOLE.APPOBJECT")

'Open the database.
objApp.OpenFile C:\My Documents\ACT\Database\ACT5demo.dbf

'Set the Preferences object.
Set objPreferences = objApp.Preferences

'Select the Move Between Records Using Act2.0 Shortcut Keys
'option (General tab).
objPreferences.UseAct20Keys = True

'Close the application.
Set objPreferences = Nothing
Set objApp = Nothing
```

UseLastDBonStartup Property

Description Gets and sets the mode of the Startup Database. Use this property to get or select the Last Opened or Named Database option in the Startup tab of the Preferences dialog box. Indicates whether ACT! starts up with the last opened database.

Object Preferences object

Syntax *object*.**UseLastDBonStartup** [= *True*|*False*]

Parameters *True*|*False* Specify True to select the Last Opened option or False to select the Named Database option. If this parameter is omitted, returnsTrue if the Last Opened option is selected or False if the Named Database option is selected.

Note: If you set this property to False, use StartupDatabase to specify the path and name of the startup database.

Value type Boolean, Read/Write

GetLastError S_NO_SECURITY, S_NO_USER, S_NOT_PRIVILEGED, S_PROPS_NOT_FOUND, S_WRITE_FAIL

See also StartupDatabase Property

Example

```
'This example specifies that the last opened database is not used
'at startup in the Preferences dialog box and accordingly in the
'ACT! application.

Dim objApp as object
Dim objPreferences as object

'Initialize the Application object.
'This starts ACT! if it is not already running.
Set objApp = CreateObject("ACTOLE.APPOBJECT")
```

```
'Open the database.
objApp.OpenFile C:\My Documents\ACT\Database\ACT5demo.dbf

'Set the Preferences object.
Set objPreferences = objApp.Preferences

'In the Startup Database group box, deselect the
'Last Opened option (Startup tab).
objPreferences.UseLastDBonStartup = False

'Close the application.
Set objPreferences = Nothing
Set objApp = Nothing
```

UseTypeahead Property

Requires ACT! 4.0 or later

Description Gets and sets the typeahead mode for entering e-mail message recipients. Use this property to get, select, or deselect (that is, enable or disable) the Use Typeahead For Entering Recipients option in the E-mail tab of the Preferences dialog box. Returns True if the Use Typeahead For Entering Recipients option is selected or False if this option is deselected. Returns False on failure.

Object Preferences object

Syntax *object*.**UseTypeahead** [= *True*|*False*]

Parameters *True*|*False* Specify True to select the Use Typeahead For Entering Recipients option or False to deselect it. Omit this optional parameter to get the setting (selected or deselected) of the UseTypeahead For Entering Recipients option.

Value type Boolean, Read/Write

GetLastError S_NO_SECURITY, S_NO_USER, S_NOT_PRIVILEGED, S_PROPS_NOT_FOUND

Example

```
Dim objApp As Object
Dim objPref As Object

'Open the database with the default database.
Set objApp = CreateObject("ACTOLE.APPOBJECT")

'Open a database.
objApp.OpenDB C:\My Documents\ACT\Database\ACT5demo.dbf

'Get the Preferences object from the Application object.
Set objPref = objApp.Preferences

'Select the typeahead option.
objPref.UseTypeahead True

'Close the Preferences object.
Set objPref = Nothing
'Close the Application object.
Set objApp = Nothing
```

WaitTime Property

Requires ACT! 4.0 or later

Description Gets and sets the value that represents the time to wait for calls to receive synchronization updates by modem. Use this property to get and set the value of the Wait For option in the Synchronization tab of the Preferences dialog box. Returns S_ERROR on failure.

Object Preferences object

Syntax *object*.**WaitTime** [= *lMinutes*]

Parameters *lMinutes* A long integer representing the number of minutes for the Wait For option. Omit this optional parameter to get the setting for the number of minutes.

The following table lists the values that are set or returned by this property.

Value	Setting	Value	Setting
0	Forever	120	2 hours
5	5 minutes	180	3 hours
10	10 minutes	360	6 hours
15	15 minutes	480	8 hours
30	30 minutes	600	10 hours
45	45 minutes	720	12 hours
60	1 hour		

Value type Long Integer, Read/Write

GetLastError S_NO_SECURITY, S_NO_USER, S_NOT_PRIVILEGED, S_PROPS_NOT_FOUND

Example

```
Dim objApp As Object
Dim objPref As Object

'Open the database with the default database.
Set objApp = CreateObject("ACTOLE.APPOBJECT")

'Open a database.
objApp.OpenDB C:\My Documents\ACT\Database\ACT5demo.dbf

'Get the Preferences object from the Application object.
Set objPref = objApp.Preferences

'Set the wait time option to 6 hours.
objPref.WaitTime 360

'Close the Preferences object.
Set objPref = Nothing
'Close the Application object.
Set objApp = Nothing
```

Methods

Name	Parameter(s)	Parameter type(s)	Return type
ClearError Method	None	*	Void
GetActivityCleanupStyle Method	None	*	Short Integer
GetAttachmentInfo Method	None	*	String/BSTR
GetCalendarIncrements Method	iType	Short Integer	Short Integer
GetDataToSynch Method	None	*	Long Integer
GetDBMaintReminderInfo Method	bEnabled lDayInterval	Boolean Long Integer	Long Integer
GetDefaultApplication Method	iType	Short Integer	String/BSTR
GetDefaultLocation Method	iType	Short Integer	String/BSTR
GetEmailInboxSettings Method	iDelete iPollTime	Short Integer Short Integer	Long Integer
GetEmailNewMsgInfo Method	None	*	String/BSTR
GetEmailSystem Method	None	*	String/BSTR
GetLastError Method	None	*	Long Integer
GetNameSettings Method	iType	Short Integer	String/BSTR
GetSchdActivityDefaults Method	iType	Short Integer	String/BSTR
GetSchdAutoRollover Method	None	*	Short Integer
GetStyle Method	iWindowType	Short Integer	String/BSTR
GetSynchScheduleInfo Method	None	*	String/BSTR
GetSynchSettings Method	None	*	String/BSTR
GetSynchUpdateInfo Method	None	*	Long Integer
SetActivityCleanupStyle Method	iStyle	Short Integer	Long Integer
SetAttachmentInfo Method	szNewInfo	String	Long Integer
SetCalendarIncrements Method	iType iIncrement	Short Integer Short Integer	Long Integer
SetDataToSynch Method	lInfo	Long Integer	Long Integer
SetDBMaintReminderInfo Method	True|False, lDayInterval	Boolean Long Integer	Void
SetDefaultApplication Method	iType szAppName	Short Integer String	Long Integer
SetDefaultLocation Method	iType,szLocation	Short Integer,String	Long Integer
SetEmailInboxSettings Method	iDelete,iPollTime	Short Integer,Short Integer	Long Integer
SetEmailNewMsgInfo Method	szInfo	String	Long Integer
SetEmailSystem Method	szSystem	String	Long Integer
SetNameSettings Method	iType,szList	Short Integer,String	Long Integer
SetSchdActivityDefaults Method	iType,szDefaults	Short Integer,String	Long Integer
SetSchdAutoRollover Method	iRollover	Short Integer	Long Integer
SetStyle Method	iWindowType,szFontName	Short Integer,String	Long Integer

Name	Parameter(s)	Parameter type(s)	Return type
SetSynchScheduleInfo Method	szInfo	String	Long Integer
SetSynchSettings Method	szInfo	String	Long Integer
SetSynchUpdateInfo Method	lInfo	Long Integer	Long Integer

ClearError Method

Requires ACT! 4.0 or later

Description Clears the last error.

Object Preferences object

Syntax *object*.**ClearError**

Return type Void

GetLastError S_NO_SECURITY, S_NO_USER, S_NOT_PRIVILEGED, S_PROPS_NOT_FOUND

See also GetLastError Method, GetActivityCleanupStyle Method

GetActivityCleanupStyle Method

Description Returns the style of representing cleared activities. Returns S_ERROR on failure.

Object Preferences object

Syntax object.GetActivityCleanupStyle

Return type Short Integer

Comments The following table lists the values that are returned by this method.

Value	Setting	Value	Setting
0	Gray cleared activities	2	Remove cleared activities
1	Strike out cleared activities		

GetLastError S_NO_SECURITY, S_NO_USER, S_NOT_PRIVILEGED, S_PROPS_NOT_FOUND

See also SetActivityCleanupStyle

GetAttachmentInfo Method

Requires ACT! 4.0 or later

Description Returns a string representing the settings for the Attaching Messages To Contacts options in the Preferences E-mail tab. Returns NULL on error.

Object Preferences object

Syntax *object*.**GetAttachmentInfo**

Return type String/BSTR

Comments The following fields are returned in order and separated by a backslash (\)

`AttachMode\Folder`

AttachMode: The following table lists the values returned for the AttachMode field.

Value	Setting
0	Ask me (the user) before saving the attached file.
1	Always save the attached file.
2	Never save the attached file.

Folder: Contains the location of the folder specified in the Messages You Have Attached To Contacts Are Stored In option.

GetLastError S_NO_SECURITY, S_NO_USER, S_NOT_PRIVILEGED, S_PROPS_NOT_FOUND

See also SetAttachmentInfo

GetCalendarIncrements Method

Description Returns the calendar increments for the daily or weekly calendar. Returns -1 on failure. Use GetLastError to get information on an error.

Object Preferences object

Syntax *object*.**GetCalendarIncrements**(*iType*)

Parameters *iType* A short integer representing the calendar.

The following table lists the values for iType.

Value	Setting
301	Weekly calendar
302	Daily calendar

Return type Short Integer

GetLastError S_NO_SECURITY, S_NO_USER, S_NOT_PRIVILEGED, S_PROPS_NOT_FOUND

Comments The returned value is 5, 10, 15, 30, 45, or 60.

See also SetCalendarIncrements Method

GetDataToSynch Method

Requires ACT! 4.0 or later

Description Returns a value that represents the settings for the Data To Synchronize options in the Preferences Synchronization tab. Returns S_ERROR on error.

Object Preferences object

Syntax *object*.**GetDataToSynch**

Return type Long Integer

Comments The following table lists the values that are returned for this method.

Value	Setting
0	Do not Synchronize Activities or Notes/Histories
1	Synchronize Activities only
2	Synchronize Notes/Histories only
3	Synchronize Activities and Notes/Histories

GetLastError S_ERROR, S_NO_SECURITY, S_NO_USER, S_NOT_PRIVILEGED, S_PROPS_NOT_FOUND

See also SetDataToSynch Method

GetDBMaintReminderInfo Method

Requires ACT! 5.0 or later

Description Gets the values for the database maintenance reminder. Returns True if the reminder is set or False if it is not set; also returns the value (in days) for the database maintenance reminder.

Object Preferences object

Syntax *object*.**GetDBMaintReminderInfo**(*bEnabed, lDayInterval*)

Parameters *bEnabled* A Boolean variable to contain the value returned for the database maintenance reminder setting.

lDayInterval A long integer variable to contain the value returned for the database maintenance reminder setting.

Return type Long Integer

See also SetDBMaintReminderInfo Method

Example

```
Dim objApp as object
Dim objPreferences as object
Dim bEnabled As Boolean
Dim lDay As Long

'Initialize the Application object.
'This starts ACT! if it is not already running.
Set objApp = CreateObject("ACTOLE.APPOBJECT")

'Open the database.
objApp.OpenFile C:\My Documents\ACT\Database\ACT5demo.dbf

'Set the Preferences object.
Set objPreferences = objApp.Preferences

objPreferences.GetDBMaintReminderInfo bEnabled, lDay
List1.AddItem " Database Maintenance On " & bEnabled &
    " set for " & lDay & " days"

'Close the application.
Set objPreferences = Nothing
Set objApp = Nothing
```

GetDefaultApplication Method

Description Returns the name of the default application to run for the word processor or fax software. Returns NULL on failure.

Object Preferences object

Syntax *object*.**GetDefaultApplication (***iType*)

Parameters *iType* A short integer representing the type of application. Specify 0 to get the default word processor or 1 or to get the default fax software.

Return type String/BSTR

GetLastError S_NO_FAX_DRVMGR, S_NO_SECURITY, S_NO_USER, S_NO_WP_DRIVER, S_NO_WP_DRVMGR, S_NOT_PRIVILEGED, S_PROPS_NOT_FOUND

See also SetDefaultApplication Method

GetDefaultLocation Method

Description Returns the default directory for the specified file type. Returns NULL on failure.

Object Preferences object

Syntax *object*.**GetDefaultLocation (***iType*)

Parameters *iType* A short integer representing the file type.

The following table lists the values for iType.

Value	Setting	Value	Setting
0	Database	6	Layout
1	Document	7	Macro
2	Document template	8	Briefcase
3	Report, label, envelope	9	Outbox
4	Synchronize	10	Spelling dictionary
5	Query	11	NetLinks (ACT! 4.0 or later)

Return type String/BSTR

GetLastError S_INVALID_INPUT, S_NO_SECURITY, S_NO_USER, S_NOT_PRIVILEGED, S_PROPS_NOT_FOUND

See also SetDefaultLocation Method

GetEmailInboxSettings Method

Description Returns the E-mail Inbox Settings options. Returns S_ERROR on failure.

Object Preferences object

Syntax *object*.**GetEmailInboxSettings**(*iDelete, iPollTime*)

Parameters *iDelete* A short integer (pointer to a short integer in Visual C++) indicating whether confirmation is required when deleting messages. Specify 0 to select this option (confirm deletions) or 1 to deselect 1.

iPollTime A short integer (pointer to a short integer in Visual C++) between 0 (zero) and 60 indicating whether to notify when new message arrives. A number greater than 0 (zero) is interpreted as the number of minutes; a value of 0 (zero) indicates that the property is cleared (not set).

Return type Long Integer

GetLastError S_NO_SECURITY, S_NO_USER, S_NOT_PRIVILEGED, S_PROPS_NOT_FOUND

See also SetEmailInboxSettings Method

Example

```
'This example gets the values of the Email inbox
'settings in the iDelete and iNotify parameters.

Dim iNotify as integer
Dim iDelete as integer
objPreferences.GetEmailInboxSettings iDelete, iNotify
List1.AddItem "GetEmailInboxSettings:"
List1.AddItem "Confirm when Deleting: " & iDelete &
    "Notify every " & iNotify & " Minutes"
```

GetEmailNewMsgInfo Method

Description Returns the signature text and the mode of creating a history record when sending a new e-mail message. Use this property to get the contents of the Signature Text and the setting of the Create History When Sent option under New Message Settings in the E-mail tab of the Preferences dialog box. Returns NULL on error.

Object Preferences object

Syntax *object*.**GetEmailNewMsgInfo**

Return type String/BSTR

Comments The returned string contains the following two fields, separated by a backslash (\):

`CreateHist\SigText`

CreateHist indicates the setting of the Create History When Sent option. The value of 1 is returned if this option is selected and 0 (zero) is returned if it is deselected.

SigText contains the Signature Text for a new message.

GetLastError S_NO_SECURITY, S_NO_USER, S_NOT_PRIVILEGED, S_PROPS_NOT_FOUND

See also SetEmailNewMsgInfo Method

Example

```
Dim objApp As Object
Dim objPref As Object

'Open the database with the default database.
Set objApp = CreateObject("ACTOLE.APPOBJECT")

'Open a database.
objApp.OpenDB C:\My Documents\ACT\Database\ACT5demo.dbf

'Get the Preferences object from the Application object.
Set objPref = objApp.Preferences

'Get the property GetEmailNewMsgInfo in a string.
strEmailMsgInfo = objPref.GetEmailNewMsgInfo

Set objPref = Nothing
Set objApp = Nothing
```

GetEmailSystem Method

Requires ACT! 4.0 or later

Description Returns a string containing the current e-mail system. Returns NULL on error or if no system is specified.

Object Preferences object

Syntax *object*.**GetEmailSystem**

Return type String/BSTR

GetLastError S_NO_SECURITY, S_NO_USER, S_NOT_PRIVILEGED, S_PROPS_NOT_FOUND

See also SetEmailSystem Method

GetLastError Method

Requires ACT! 4.0 or later

Description Returns a long integer representing the last error code for the object. For more information, see Error codes.

Object Preferences object

Syntax *object*.**GetLastError**

GetLastError S_NO_SECURITY, S_NO_USER, S_NOT_PRIVILEGED, S_PROPS_NOT_FOUND

Return type Long Integer

GetNameSettings Method

Description Returns a string containing the currently defined first-name prefixes, last-name prefixes, and last-name suffixes. Returns NULL on error.

Object Preferences object

Syntax *object*.**GetNameSettings**(*iType*)

Parameters *iType* A short integer. (Currently ignored.)

Return type String/BSTR

GetLastError S_NO_SECURITY, S_NO_USER, S_NOT_PRIVILEGED, S_PROPS_NOT_FOUND

Comments The returned string contains the following three fields, separated by a backslash (\):

```
FirstNamePrefixes\LastNamePrefixes\LastNameSuffixes
```

Within each field, prefixes or suffixes are separated by commas.

See also SetNameSettings

GetSchdActivityDefaults Method

Description Returns the schedule activity default settings. Returns NULL on failure.

Object Preferences object

Syntax *object*.**GetSchdActivityDefaults**(*iType*)

Parameters *iType* A short integer representing the activity type for which the defaults are required.

The following table lists the values for iType.

Value	Setting	Value	Setting
0	Calls	2	To-dos
1	Meetings		

Return type String/BSTR

GetLastError S_INVALID_INPUT, S_NO_SECURITY, S_NO_USER, S_NOT_PRIVILEGED, S_PROPS_NOT_FOUND

Comments For the specified activity, the returned string contains the following six fields, separated by a backslash (\):

```
Priority\AlarmLeadTime\Duration\DefaultToTimeless\SetAlarm\Popup
  Fields
```

Priority: The following table lists the values returned for the Priority field.

Value	Setting	Value	Setting
0	High	2	Low
1	Medium		

AlarmLeadTime: The following table lists the values returned for the AlarmLeadTime field.

Value	Setting	Value	Setting
0	0 minutes	7	2 hours
1	5 minutes	8	3 hours
2	10 minutes	9	8 hours
3	15 minutes	10	1 day

Value	Setting	Value	Setting
4	30 minutes	11	5 days
5	45 minutes	12	30 days
6	1 hour		

Duration: The following table lists the values returned for the Duration field.

Value	Setting	Value	Setting
0	0 minutes	7	2 hours
1	5 minutes	8	3 hours
2	10 minutes	9	8 hours
3	15 minutes	10	1 day
4	30 minutes	11	5 days
5	45 minutes	12	30 days
6	1 hour		

DefaultToTimeless: DefaultToTimeless indicates the setting of the Default x To Timeless option. The value of 1 is returned if this option is selected and 0 (zero) is returned if it is deselected.

SetAlarm: SetAlarm indicates the setting of the Set Alarm For x option. The value of 1 is returned if this option is selected and 0 (zero) is returned if it is deselected.

PopupFields: A value in a range from 0 to 31 that indicates which fields are displayed in the Schedule Activity dialog box. Popup field preferences are provided only in ACT! 4.0 or later.

PopupFields is created by ORing the values in the following table.

Value	Setting	Value	Setting
1	Date	8	Regarding
2	Time	16	Alarm Lead Time
4	Duration		

See also SetSchdActivityDefaults Method

GetSchdAutoRollover Method

Requires ACT! 4.0 or earlier.

Description Returns the automatic rollover settings. The returned integer can be used to extract the rollover settings for calls, meetings, and to-do's by ANDing it with appropriate values. Returns S_ERROR on failure.

Object Preferences object

Syntax *object*.**GetSchdAutoRollover**

Return type Short Integer

GetLastError S_NO_SECURITY, S_NO_USER, S_NOT_PRIVILEGED, S_PROPS_NOT_FOUND

Comments The returned value contains a total that represents all selected activity types. For example, a returned value of "5" indicates that the Calls and To-do's options are selected.

The following table lists the values that are added to get the total value that is returned for this method.

Value	Setting	Value	Setting
1	Calls	4	To-do's
2	Meetings		

See also SetSchdAutoRollover Method

Example

```
Set objPref = objApp.Preferences
Dim strType As String
Dim i as integer
strType = ""
i = objPref.GetSchdAutoRollover

'Now And the result to get the types
If i And 1 Then strType = "Calls "
If i And 2 Then strType = strType & "Meetings "
If i And 4 Then strType = strType & "Todo "
List1.AddItem strType
```

GetStyle Method

Description Returns the name of the font used in the specified window. Returns NULL on error.

Object Preferences object

Syntax object.**GetStyle**(iWindowType)

Parameters iWindowType A short integer representing the window type.

The following table lists the values for this parameter:

Value	Setting	Value	Setting
0	Contact List	5	Weekly calendar
1	Notes/History tab	6	Monthly calendar
2	Activities tab	7	E-mail
3	Task List	8	Contacts tab
4	Daily calendar	9	Groups view

Return type String/BSTR

GetLastError S_NO_SECURITY, S_NO_USER, S_NOT_PRIVILEGED, S_PROPS_NOT_FOUND

See also SetDefaultLocation Method, SetStyle Method

GetSynchScheduleInfo Method

Requires ACT! 4.0 or later

Description Returns the settings for the auto synchronization schedule. Returns NULL on error.

Note: Use this method instead of GetSynchSchedule in ACT! 4.0 or later.

Object Preferences object

Syntax object.GetSynchScheduleInfo

Return type String/BSTR

GetLastError S_NO_SECURITY, S_NO_USER, S_NOT_PRIVILEGED, S_PROPS_NOT_FOUND

Comments The returned string contains the following two fields, separated by a backslash (\):

`SynchDays\ SynchTime…`

SynchDays A value between 0 and 128 representing the days of the week on which synchronization is to be performed. The value must be ANDed with specific defines representing the week bit set.

The following table lists the values for days of a week returned for the SynchDays field.

Value	Day of week	Value	Day of week
1	Sunday	16	Thursday
2	Monday	32	Friday
4	Tuesday	64	Saturday
8	Wednesday		

SynchTime Synchronization time(s) in Windows Regional Settings Time style (hh:mm AM|PM) format. The synchronization times are returned in the following format:

`\SynchTime\SynchTime\SynchTime...`

See also GetSynchScheduleInfo Method, GetSynchSettings Method, SetSynchScheduleInfo Method, SetSynchSettings Method, SetSynchUpdateInfo Method

Example

```
Dim objApp As Object
Dim objPref As Object
Dim strSynchScheduleInfo

'Open the database with the default database.
Set objApp = CreateObject("ACTOLE.APPOBJECT")

'Open a database.
objApp.OpenDB C:\My Documents\ACT\Database\ACT5demo.dbf

'Get the Preferences object from the Application object.
Set objPref = objApp.Preferences

'Get the property GetSynchScheduleInfo in a string.
strSynchScheduleInfo = objPref.GetSynchScheduleInfo

'Close the Preferences object.
Set objPref = Nothing
'Close the Application object.
Set objApp = Nothing
```

GetSynchSettings Method

Description Returns the values for the synchronization settings. Returns NULL on error.

Object Preferences object

Syntax *object.***GetSynchSettings**

Return type String/BSTR

GetLastError S_NO_SECURITY, S_NO_USER, S_NOT_PRIVILEGED, S_PROPS_NOT_FOUND

Comments The returned string contains the following two fields, separated by a backslash (\):

`ReminderDays\PurgeFreq`

ReminderDays Number of days before displaying reminder (1 to 1000). Value is 0 (zero) if synchronization reminders are disabled.

PurgeFreq Number of days after which the transaction log is to be purged (1 to 400).

See also GetSynchScheduleInfo Method, SetSynchScheduleInfo Method, SetSynchSettings Method

GetSynchUpdateInfo Method

Requires ACT! 4.0 or later

Description Returns a value that represents the settings for the When Synchronizing options in the Preferences Synchronization tab. Returns S_ERROR on error.

Object Preferences object

Syntax *object*.**GetSynchUpdateInfo**

Return type Long Integer

GetLastError S_ERROR, S_NO_SECURITY, S_NO_USER, S_NOT_PRIVILEGED, S_PROPS_NOT_FOUND

Comments The following table lists the values that are returned for this property.

Value	Setting
0	Do not Send or Receive Updates
1	Send Updates only
2	Receive Updates only
3	Send and Receive Updates

See also SetSynchUpdateInfo Method

SetActivityCleanupStyle Method

Description Sets a value representing the style of representing cleared activities. Returns S_ERROR on failure.

Object Preferences object

Syntax *object*.**SetActivityCleanupStyle**(*iStyle*)

Parameters *iStyle* A short integer representing the style for cleared activities. Specify 0 to gray cleared activities or 1 to strike out cleared activities.

Return type Long Integer

GetLastError S_INVALID_INPUT, S_NO_SECURITY, S_NO_USER, S_NOT_PRIVILEGED, S_PROPS_NOT_FOUND, S_WRITE_FAIL

See also GetActivityCleanupStyle Method

Example

```
'This example specifies that cleared activities should be grayed in
'the Preferences dialog box and accordingly in the ACT! application.

Dim objApp as object
Dim objPreferences as object

'Initialize the Application object.
'This starts ACT! if it is not already running.
Set objApp = CreateObject("ACTOLE.APPOBJECT")

'Open the database.
objApp.OpenFile C:\My Documents\ACT\Database\ACT5demo.dbf
'Set the Preferences object.
Set objPreferences = objApp.Preferences
```

```
'Select the Gray option in the When Clearing Activities
'group box (Scheduling tab).
objPreferences.SetActivityCleanupStyle 0

'Close the application.
Set objPreferences = Nothing
Set objApp = Nothing
```

SetAttachmentInfo Method

Requires ACT! 4.0 or later

Description Sets values representing the settings for the Attaching Messages To Contacts options in the Preferences E-mail tab. Returns S_ERROR on error.

Object Preferences object

Syntax *object*.**SetAttachmentInfo**(*szNewInfo*)

Parameters *szNewInfo* The following fields are specified in szNewInfo in order and separated by a backslash (\)

`AttachMode\Folder`

AttachMode:

The following table lists the values for the AttachMode field.

Value	Setting
0	Ask me (the user) before saving the attached file.
1	Always save the attached file.
2	Never save the attached file.

Folder:

Contains the location of the folder specified in the Messages You Have Attached To Contacts Are Stored In option. An error is returned if the specified folder does not exist.

Return type Long Integer

GetLastError S_NO_SECURITY, S_NO_USER, S_NOT_PRIVILEGED, S_PROPS_NOT_FOUND, S_WRITE_FAIL

See also GetAttachmentInfo Method

Example

```
'This example selects the Always save the attached file option for
'When attaching messages with file attachments and the folder name
'in E-mail preferences.

Dim objApp as object
Dim objPreferences as object

'Initialize the Application object.
'This starts ACT! if it is not already running.
Set objApp = CreateObject("ACTOLE.APPOBJECT")

'Open the database.
objApp.OpenFile C:\My Documents\ACT\Database\ACT5demo.dbf
'Set the Preferences object.
Set objPreferences = objApp.Preferences

'Selects the Always save the attached file option and
```

```
'the Folder name as c:\Email.
objPreferences.SetAttachmentInfo "1\c:\Email"

'Close the application.
Set objPreferences = Nothing
Set objApp = Nothing
```

SetCalendarIncrements Method

Description Sets values representing the settings of calendar increments for the daily or weekly calendar. Returns S_ERROR on failure.

Object Preferences object

Syntax *object*.**SetCalendarIncrements**(*iType, iIncrement*)

Parameters *iType* A short integer representing the calendar. The following table lists the values for iType.

Value	Setting
301	Weekly calendar
302	Daily calendar

iIncrement A short integer representing the increment in minutes. iIncrement can be 5, 10, 15, 30, 45, or 60. Any other value is converted to 5.

Return type Long Integer

GetLastError S_NO_SECURITY, S_NO_USER, S_NOT_PRIVILEGED, S_PROPS_NOT_FOUND, S_WRITE_FAIL

See also GetCalendarIncrements Method

Example

```
'This example sets weekly calendar increments in the Preferences
'dialog box and accordingly in the ACT! application.

Dim objApp as object
Dim objPreferences as object

'Initialize the Application object.
'This starts ACT! if it is not already running.
Set objApp = CreateObject("ACTOLE.APPOBJECT")

'Open the database.
objApp.OpenFile C:\My Documents\ACT\Database\ACT5demo.dbf

'Set the Preferences object.
Set objPreferences = objApp.Preferences

'Set weekly calendar increments to 1 hour (Calendars tab).
objPreferences.SetCalendarIncrements 301, 60

'Close the application.
Set objPreferences = Nothing
Set objApp = Nothing
```

SetDataToSynch Method

Requires ACT! 4.0 or later

Description Sets a value that selects or deselects the Data To Synchronize options in the Preferences Synchronization tab. Returns S_ERROR on error.

Object Preferences object

Syntax *object*.**SetDataToSynch**(*lInfo*)

Parameters *lInfo* A long integer in the range from 0 to 3 that represents the settings for the Data To Synchronize options in the Preferences Synchronization tab.

The following table lists the values for this parameter.

Value	Setting
0	Do not Synchronize Activities or Notes/Histories
1	Synchronize Activities only
2	Synchronize Notes/Histories only
3	Synchronize Activities and Notes/Histories

Return type Long Integer

GetLastError S_ERROR, S_INVALID_INPUT, S_NO_SECURITY, S_NO_USER, S_NOT_PRIVILEGED, S_PROPS_NOT_FOUND

See also GetDataToSynch Method

Example

```
'This example selects only the Notes/Histories option for Data to
'synchronize in Synchronization preferences.

Dim objApp as object
Dim objPreferences as object

'Initialize the Application object.
'This starts ACT! if it is not already running.
Set objApp = CreateObject("ACTOLE.APPOBJECT")

'Open the database.
objApp.OpenFile C:\My Documents\ACT\Database\ACT5demo.dbf

'Set the Preferences object.
Set objPreferences = objApp.Preferences

'Checks the Notes/History checkbox in the Data to be Synchronized.
objPreferences.SetDataToSynch 2

'Close the application.
Set objPreferences = Nothing
Set objApp = Nothing
```

SetDBMaintReminderInfo Method

Requires ACT! 5.0 or later

Description Sets the values for the database maintenance reminder.

Object Preferences object

Syntax *object*.**SetDBMaintReminderInfo** (*True|False, lDayInterval*)

Parameters *True|False* Specify True to select the database maintenance reminder setting or False to deselect the setting.

lDayInterval A long integer specifying the value (in days) for the database maintenance reminder, in a range between 1 and 365. This value is not validated if True|False is set to False.

Return type Void

See also GetDBMaintReminderInfo Method

Example

```
Dim objApp as object
Dim objPreferences as object

'Initialize the Application object.
'This starts ACT! if it is not already running.
Set objApp = CreateObject("ACTOLE.APPOBJECT")

'Open the database.
objApp.OpenFile C:\My Documents\ACT\Database\ACT5demo.dbf

'Set the Preferences object.
Set objPreferences = objApp.Preferences

'Set the database maintenance reminder to be set on for every 15 days.
objPreferences.SetDBMaintReminderInfo True, 15

'Close the application.
Set objPreferences = Nothing
Set objApp = Nothing
```

SetDefaultApplication Method

Description Sets the default application to be used for the specified type. Returns S_ERROR on failure.

Object Preferences object

Syntax *object*.**SetDefaultApplication (***iType, szAppName*)

Parameters *iType* A short integer representing the type of application. Specify 0 to set the default word processor or 1 or to set the default fax software.

szAppName A string representing the name of the application, such as "ACT! Word Processor" or "Microsoft Word 95 - 2000."

Return type Long Integer

GetLastError S_INVALID_INPUT, S_NO_FAX_DRIVER, S_NO_FAX_DRVMGR, S_NO_SECURITY, S_NO_USER, S_NO_WP_DRIVER, S_NO_WP_DRVMGR, S_NOT_PRIVILEGED, S_PROPS_NOT_FOUND

See also GetDefaultApplication Method

Example

```
'This example specifies that the default word processor is the
'ACT! word processor in the Preferences dialog box and
'accordingly in the ACT! application.

Dim objApp as object
Dim objPreferences as object
'Initialize the Application object.
'This starts ACT! if it is not already running.
Set objApp = CreateObject("ACTOLE.APPOBJECT")
```

```
'Open the database.
objApp.OpenFile C:\My Documents\ACT\Database\ACT5demo.dbf

'Set the Preferences object.
Set objPreferences = objApp.Preferences

'Select the ACT! Word Processor in the Word Processor list in the
'Default Applications group box (General tab).
objPreferences.SetDefaultApplication 1, "ACT! Word Processor"

'Close the application.
Set objPreferences = Nothing
Set objApp = Nothing
```

SetDefaultLocation Method

Description Sets the default directory location for the specified file type. Returns S_ERROR if the specified file does not exist.

Object Preferences object

Syntax *object*.**SetDefaultLocation (***iType, szLocation)*

Parameters *iType* A short integer representing the file type. The following table lists the values for this parameter.

Value	Setting	Value	Setting
0	Database	6	Layout
1	Document	7	Macro
2	Document template	8	Briefcase
3	Report, label, envelope	9	Outbox
4	Synchronize	10	Spelling dictionary
5	Query	12	NetLinks (ACT! 4.0 or later)

szLocation A string representing the default location of the files of iType type.

Return type Long Integer

GetLastError S_INVALID_INPUT, S_NO_SECURITY, S_NO_USER, S_NOT_FOUND, S_NOT_PRIVILEGED, S_PROPS_NOT_FOUND, S_WRITE_FAIL

See also GetDefaultLocation Method

Example

```
'This example specifies the default location for documents in the
'Preferences dialog box and accordingly in the ACT! application.

Dim objApp as object
Dim objPreferences as object

'Initialize the Application object.
'This starts ACT! if it is not already running.
Set objApp = CreateObject("ACTOLE.APPOBJECT")

'Open the database.
objApp.OpenFile C:\My Documents\ACT\Database\ACT5demo.dbf
```

```
'Set the Preferences object.
Set objPreferences = objApp.Preferences

'Set the default location for documents to C:\MY DOCUMENTS (General tab).
objPreferences.SetDefaultLocation 2, "C:\MY DOCUMENTS"

'Close the application.
Set objPreferences = Nothing
Set objApp = Nothing
```

SetEmailInboxSettings Method

Description Sets the E-mail Inbox Settings options.

Object Preferences object

Syntax *object*.**SetEmailInboxSettings**(*iDelete, iPollTime*)

Parameters *iDelete* A short integer specifying the setting of the Confirm When Deleting Message(s) option in the E-mail tab for preferences. Specify a value of 1 to select this option or 0 (zero) to deselect it.

Note: Although still required, the iDelete parameter is obsolete in ACT! 6.0. Deleted e-mail is sent to the Deleted Items folder and confirmation has been eliminated as a result.

iPollTime A short integer in the range between 0 (zero) and 60 specifying the setting of the When Connected Notify Me Of New Mail Every xx Minutes option in the E-mail tab for preferences. A value greater than 0 (zero) indicates the number of minutes; a value of 0 (zero) indicates that the option is deselected.

Return type Long Integer

GetLastError S_NO_SECURITY, S_NO_USER, S_NOT_PRIVILEGED, S_PROPS_NOT_FOUND, S_WRITE_FAIL

See also GetEmailInboxSettings Method

Example

```
'This example specifies e-mail settings in the Preferences
'dialog box and accordingly in the application.

Dim objApp as object
Dim objPreferences as object

'Initialize the Application object.
'This starts ACT! if it is not already running.
Set objApp = CreateObject("ACTOLE.APPOBJECT")

'Open the database.
objApp.OpenFile C:\My Documents\ACT\Database\ACT5demo.dbf

'Set the Preferences object.
Set objPreferences = objApp.Preferences

'Select the Confirm When Deleting Message(s) option and the Notify Me
'of New E-mail option and set it to 5 minutes (E-mail tab).
objPreferences.SetEmailInboxSettings 1, 5

'Close the application.
Set objPreferences = Nothing
Set objApp = Nothing
```

SetEmailNewMsgInfo Method

Description Sets the signature text and the mode of creating a history record when sending a new e-mail message. Use this property to set the Signature Text and the setting of the Create History When Sent option under New Message Settings in the E-mail tab of the Preferences dialog box. Returns S_ERROR on failure.

Object Preferences object

Syntax *object*.**SetEmailNewMsgInfo**(*szInfo*)

Parameters *szInfo* A string containing the information.

Return type Long Integer

GetLastError S_NO_SECURITY, S_NO_USER, S_NOT_PRIVILEGED, S_PROPS_NOT_FOUND, S_WRITE_FAIL

Comments The string contains the following two fields, separated by a backslash (\):

`CreateHist\SigText`

CreateHist specifies the setting of the Create History When Sent option. Specify 1 to select this option or 0 (zero) to deselect it.

Note: Although still required, the CreateHist field is obsolete in ACT! 6.0. History is created based on the user's selection when sending e-mail. By default, history items are based only on the subject line.

SigText specifies Signature Text for a new message.

See also GetEmailNewMsgInfo Method

Example

```
'This example specifies new e-mail settings in the Preferences
'dialog box and accordingly in the ACT! application.

Dim objApp as object
Dim objPreferences as object

'Initialize the Application object.
'This starts ACT! if it is not already running.
Set objApp = CreateObject("ACTOLE.APPOBJECT")

'Open the database.
objApp.OpenFile C:\My Documents\ACT\Database\ACT5demo.dbf

'Set the Preferences object.
Set objPreferences = objApp.Preferences

'Select the Create History When Sent option in the New Message Settings
'group box (E-mail tab). Display "Chris Huffman " as the signature text.
objPreferences.SetEmailNewMsgInfo "1\Chris Huffman "

'Close the application.
Set objPreferences = Nothing
Set objApp = Nothing
```

SetEmailSystem Method

Requires ACT! 4.0 or later

Description Sets the current E-mail system. Returns S_ERROR on failure or if the specified system is not available.

Object Preferences object

Syntax *object*.**SetEmailSystem** (*szSystem*)

Parameters *szSystem* A string that specifies the name of the mail system.

Return type Long Integer

GetLastError S_NO_EMAIL, S_NO_SECURITY, S_NO_USER, S_NOT_PRIVILEGED, S_PROPS_NOT_FOUND, S_WRITE_FAIL

See also GetEmailSystem

Example

```
'This example sets cc:Mail as the current E-mail system in
'E-mail preferences

Dim objApp as object
Dim objPreferences as object

'Initialize the Application object.
'This starts ACT! if it is not already running.
Set objApp = CreateObject("ACTOLE.APPOBJECT")

'Open the database.
objApp.OpenFile C:\My Documents\ACT\Database\ACT5demo.dbf

'Set the Preferences object.
Set objPreferences = objApp.Preferences

'Sets the Email-System to send email to Contacts as cc:Mail
'in the Preferences dialog box in the ACT! application
objPreferences.SetEmailSystem "cc:Mail"

'Close the application.
Set objPreferences = Nothing
Set objApp = Nothing
```

SetNameSettings Method

Description Sets first-name prefixes, last-name prefixes, or last-name suffixes. The prefixes or suffixes specified with this method completely overwrite any existing prefixes or suffixes (displayed in the list box of the Names tab of the Preferences dialog box). Returns S_ERROR on failure.

Object Preferences object

Syntax *object*.**SetNameSettings** *(iType, szList)*

Parameters *iType* A short integer indicating the type of prefix or suffix being specified. The following table lists the values for this parameter.

Value	Setting	Value	Setting
1	First-name prefixes	3	Last-name suffixes
2	Last-name prefixes		

szList A string containing the prefixes or suffixes, separated by commas.

Return type Long Integer

GetLastError S_NO_SECURITY, S_NO_USER, S_NOT_PRIVILEGED, S_PROPS_NOT_FOUND, S_WRITE_FAIL

See also GetNameSettings Method

Example

```
'This example specifies three first-name prefixes, overwriting
'any existing first-name prefixes in the file.
Dim objApp as object
Dim objPreferences as object

'Initialize the Application object.
'This starts ACT! if it is not already running.
Set objApp = CreateObject("ACTOLE.APPOBJECT")

'Open the database.
objApp.OpenFile C:\My Documents\ACT\Database\ACT5demo.dbf

'Set the Preferences object.
Set objPreferences = objApp.Preferences

objPreferences.SetNameSettings 1, "Col., Lt., Maj."

'Close the application.
Set objPreferences = Nothing
Set objApp = Nothing
```

SetSchdActivityDefaults Method

Description Sets the scheduling preferences default options. Returns S_ERROR on failure.

Note: Scheduling popup field preferences require ACT! 4.0 or later.

Object Preferences object

Syntax *object*.**SetSchdActivityDefaults** (*iType, szDefaults*)

Parameters *iType* A short integer representing the activity type whose defaults are to be set. The following table lists the values for this parameter.

Value	Setting	Value	Setting
0	Calls	2	To-do's
1	Meetings		

szDefaults A string containing the default values for the fields. The following fields are specified in szDefaults in order and separated by a backslash (\) for the specified type of activity:

`Priority\AlarmLeadTime\Duration\DefaultToTimeless\SetAlarm\Popup Fields`

Priority: The following table lists the values for this field.

Value	Setting	Value	Setting
0	High	2	Low
1	Medium		

AlarmLeadTime: The following table lists the values for this field.

Value	Setting	Value	Setting
0	0 minutes	7	2 hours
1	5 minutes	8	3 hours
2	10 minutes	9	8 hours

Value	Setting	Value	Setting
3	15 minutes	10	1 day
4	30 minutes	11	5 days
5	45 minutes	12	30 days
6	1 hour		

Duration: The following table lists the values for this field.

Value	Setting	Value	Setting
0	0 minutes	7	2 hours
1	5 minutes	8	3 hours
2	10 minutes	9	8 hours
3	15 minutes	10	1 day
4	30 minutes	11	5 days
5	45 minutes	12	30 days
6	1 hour		

DefaultToTimeless: DefaultToTimeless represents the setting of the Default To Timeless option. Specify a value of 1 to select this option or a value of 0 (zero) to deselect it.

SetAlarm: SetAlarm represents the setting of the Set Alarm option. Specify a value of 1 to select this option or a value of 0 (zero) to deselect it.

PopupFields: A value in a range from 0 to 31 that represents which fields are displayed in the Schedule Activity dialog box. Popup field preferences are provided only in ACT! 4.0 or later. PopupFields is created by ORing the values in the following table.

Value	Setting	Value	Setting
1	Date	8	Regarding
2	Time	16	Alarm Lead Time
4	Duration		

Return type Long Integer

GetLastError S_INVALID_INPUT, S_NO_SECURITY, S_NO_USER, S_NOT_PRIVILEGED, S_PROPS_NOT_FOUND, S_WRITE_FAIL

See also GetSchdActivityDefaults Method

Example

```
'This example specifies default settings for to-do's in the
'Preferences dialog box and accordingly in the ACT! application.

Dim objApp as object
Dim objPreferences as object

'Initialize the Application object.
'This starts ACT! if it is not already running.
Set objApp = CreateObject("ACTOLE.APPOBJECT")

'Open the database.
objApp.OpenFile C:\My Documents\ACT\Database\ACT5demo.dbf
```

```
'Set the Preferences object.
Set objPreferences = objApp.Preferences

'For to-do's, select: Priority Low, Alarm Lead Time 30 minutes,
'Duration 30 minutes, deselect the Default to Timeless option, and
'select the Set Alarm option (Scheduling tab).
Dim sPop As String
sPop = Trim(str(4 Or 16))
'The Duration and Lead Time options should be selected.
objPreferences.SetSchdActivityDefaults(2, "2\4\4\2\1\" & sPop)

'Close the application.
Set objPreferences = Nothing
Set objApp = Nothing
```

SetSchdAutoRollover Method

Requires ACT! 4.0 or earlier.

Description Sets the automatic rollover options. Returns S_ERROR on failure.

Object Preferences object

Syntax *object*.**SetSchdAutoRollover** (*iRollover*)

Parameters *iRollover* A short integer indicating the rollover settings for calls, meetings, and to-do's. iRollover is created by ORing the values in the following table for calls, meetings, and to-do's.

Value	Setting	Value	Setting
1	Calls	4	To-do's
2	Meetings		

Return type Long Integer

GetLastError S_NO_SECURITY, S_NO_USER, S_NOT_PRIVILEGED, S_PROPS_NOT_FOUND, S_WRITE_FAIL

See also GetSchdAutoRollover Method

Example

```
'This example specifies that calls and to-do's should be rolled over in
'the Preferences dialog box and accordingly in the ACT! application.
Dim objApp as object
Dim objPreferences as object

'Initialize the Application object.
'This starts ACT! if it is not already running.
Set objApp = CreateObject("ACTOLE.APPOBJECT")
'Open the database.
objApp.OpenFile C:\My Documents\ACT\Database\ACT5demo.dbf

'Set the Preferences object.
Set objPreferences = objApp.Preferences
'Select the Calls and To-do's options in the
'Automatically Roll Over group box (Scheduling tab)
objPreferences.SetSchdAutoRollover 1 Or 4

'Close the application.
Set objPreferences = Nothing
Set objApp = Nothing
```

SetStyle Method

Description Sets the name of the font to be used for the specified window.

Object Preferences object

Syntax *object*.**SetStyle** (*iWindowType, szFontName*)

Parameters *iWindowType* A short integer representing the window type. The following table lists the values for this parameter.

Value	Setting	Value	Setting
0	Contact List	5	Weekly calendar
1	Notes/History tab	6	Monthly calendar
2	Activities tab	7	E-mail
3	Task List	8	Contacts tab
4	Daily calendar	9	Groups view

szFontName A string indicating the name of the font to be used for the window.

Return type Long Integer

GetLastError S_INVALID_INPUT, S_NO_SECURITY, S_NO_USER, S_NOT_PRIVILEGED, S_PROPS_NOT_FOUND, S_WRITE_FAIL

See also GetStyle Method

Example

```
'This example specifies the font to be used in the Task List in the
'Preferences dialog box and accordingly in the ACT! application.

Dim objApp as object
Dim objPreferences as object

'Initialize the Application object.
'This starts ACT! if it is not already running.
Set objApp = CreateObject("ACTOLE.APPOBJECT")

'Open the database.
objApp.OpenFile C:\My Documents\ACT\Database\ACT5demo.dbf

'Set the Preferences object.
Set objPreferences = objApp.Preferences

'Select MS Serif for the Contact List font (Colors and Fonts tab).
objPreferences.SetStyle 0, "MS Serif"

'Close the application.
Set objPreferences = Nothing
Set objApp = Nothing
```

SetSynchScheduleInfo Method

Requires ACT! 4.0 or later

Description Sets values for the automatic synchronization schedule. Returns S_ERROR on error.

Note: Use this method instead of SetSynchSchedule in ACT! 4.0 or later.

Object Preferences object

Syntax *object*.**SetSynchScheduleInfo** (*szInfo*)

Parameters *szInfo* A string representing the information. The string contains the following two fields, separated by a backslash (\):

```
SynchDays\SynchTime
```

SynchDays: A value between 0 and 128 representing the days of the week synchronization is to be performed. The value must be ANDed with specific defines representing the week bit set. Use the following values for days of a week:

Value	Day of week	Value	Day of week
1	Sunday	16	Thursday
2	Monday	32	Friday
4	Tuesday	64	Saturday
8	Wednesday		

SynchTime: Synchronization time(s) in Windows Regional Settings Time style (hh:mm AM|PM) format. Specify synchronization times in the following format:

```
\SynchTime\SynchTime\SynchTime...
```

Return type Long Integer

GetLastError S_INVALID_INPUT, S_NO_SECURITY, S_NO_USER, S_NOT_PRIVILEGED, S_PROPS_NOT_FOUND, S_WRITE_FAIL

See also GetSynchScheduleInfo Method, GetSynchSettings Method, SetSynchSettings Method

Example

```
Set objPref = objApp.Preferences

'Set the synchronization schedule as Sunday, Thursday, and Friday,
'at 10:15 AM, 2:30 PM, and 4:45 PM.
Dim days As Integer
days = 1 Or 16 Or 32
objPref.SetSynchScheduleInfo days & "\10:15 AM\2:30 PM\4:45 PM"

Set objPref = Nothing
```

SetSynchSettings Method

Description Sets the values for the synchronization settings. Returns S_ERROR on error.

Object Preferences object

Syntax *object*.**SetSynchSettings** (*szInfo*)

Parameters *szInfo* A string containing the synchronization settings.

Return type Long Integer

GetLastError S_NO_SECURITY, S_NO_USER, S_NOT_PRIVILEGED, S_PROPS_NOT_FOUND, S_WRITE_FAIL

Comments *szInfo* must be a string containing the following three fields, separated by a backslash (\):

```
SynchReminder\ReminderDays\PurgeFreq
```

*SynchReminder*0 (zero) or 1 representing whether the user is to be reminded to perform synchronization.

*ReminderDays*Number of days before displaying reminder (1 to 1000).

PurgeFreq Number of days after which the transaction log is to be purged (1 to 400).

See also GetSynchScheduleInfo Method, GetSynchSettings Method, SetSynchScheduleInfo Method.

SetSynchUpdateInfo Method

Requires ACT! 4.0 or later

Description Sets a value that selects or deselects the When Synchronizing options in the Preferences Synchronization tab. Returns S_ERROR on error.

Object Preferences object

Syntax *object*.**SetSynchUpdateInfo**(*lInfo*)

Parameters *lInfo* A long integer in the range from 0 to 3 that represents the settings for the Send Updates and Receive Updates options in the Preferences Synchronization tab.

The following table lists the values for this parameter.

Value	Setting
0	Do not Send or Receive Updates
1	Send Updates only
2	Receive Updates only
3	Send and Receive Updates

Return type Long Integer

GetLastError S_INVALID_INPUT, S_NO_SECURITY, S_NO_USER, S_NOT_PRIVILEGED, S_PROPS_NOT_FOUND

See also GetSynchUpdateInfo Method

Example

```
'This example selects only the Send Updates option for When synchronizing
'in Synchronization preferences.

Dim objApp as object
Dim objPreferences as object

'Initialize the Application object.
'This starts ACT! if it is not already running.
Set objApp = CreateObject("ACTOLE.APPOBJECT")

'Open the database.
objApp.OpenFile C:\My Documents\ACT\Database\ACT5demo.dbf

'Set the Preferences object.
Set objPreferences = objApp.Preferences

'Checks the Send Updates Only checkbox in When Synchronizing.
objPreferences.SetSynchUpdateInfo 1

'Close the application.
Set objPreferences = Nothing
Set objApp = Nothing
```

TaskListView object

The TaskListView object provides an interface to use the Task List view in the ACT! application. The following methods apply only to the TaskListView object. See Common properties and methods for properties and additional methods that apply to this object.

Methods

Name	Parameter(s)	Parameter type(s)	Return type
AddNewActivityEx Method	szContactID… szDateTime iType iTimeless szRegarding lDuration	String String Short Integer Short Integer String Long Integer	String/BSTR
GetGrid Method	None	*	Object/LPDISPATCH

AddNewActivityEx Method

Requires ACT! 4.0 or later

Description Adds a new activity for the specified contact to the grid in theTask List view. Activity defaults set in the Scheduling tab of the Preferences dialog box are used to create the new activity. Use GetFilter Method and SetFilter Method in the Grid object before using this method. Returns S_ERROR on failure.

Object TaskListView object

Syntax *object*.**AddNewActivityEx**(*szContactID…, szDateTime, iType, iTimeless, szRegarding, lDuration*)

Parameters *szContactID* A string representing the Unique ID(s) of contacts with which the new activity will be associated. If a NULL value is specified for this required parameter, the new activity will be associated with the default of the current database user.

Note: Specify multiple Unique IDs as a continuous string of 12?character values, with no delimiters between the Unique ID values.

szDateTime A string representing the starting date and time of the activity, formatted in Windows Regional Settings Short Date style and Time style.

iType A short integer indicating the type of activity to be added. IType must be one of the following values:

Value	Activity type	Value	Activity type
0	Call	2	To-do
1	Meeting		

iTimeless A short integer indicating the timeless status of the activity to be added. Specify 0 for not timeless or 1 for timeless.

szRegarding A string representing the description of the activity.

lDuration A long integer representing the duration in minutes for the activity.

Return type String/BSTR

GetLastError S_ACTIVITY_INIT_FAIL, S_CON_DOC, S_EEDIT_FAIL, S_INVALID_INPUT, S_MEM_ERROR, S_NO_DALLIST_VIEW, S_PROPS_NOT_FOUND

Comments If any Unique ID specified in szContactID is invalid, this method returns S_OK. GetLastError returns S_INVALID_INPUT.

Return type Long Integer

Example

```
'This example adds an activity through the Task List.
'Create the TaskListView object.
Set objTask = objViews.Create(4, "TL")
'Get the Grid object.
Set objTaskGrid = objTask.GetGrid
List1.AddItem objTaskGrid.GetCurrentRow

List1.AddItem "There are " & objTaskGrid.GetRowCount & " Activities"

Dim x As Long
Dim y, z As Variant
'Get the filter values.
objTaskGrid.GetFilter x, y, z
j = 0 Or 1 Or 2 Or 8 Or 4 Or 64 Or 128 Or 16 Or 32
'Set filter to select all priorities, all types, cleared activities,
'all users, and all dates.
objTaskGrid.SetFilter j, 1, ""
'Add a new activity.
uid = objTask.AddNewActivityEx(sContact, "1/31/98 8:29PM",
    activitytype_meeting, 0, "Test ABC", "15")
RN = objTaskGrid.GetRowNumber(uid)
'Set the activity priority.
objTaskGrid.SetField AF_Priority, RN, activitypriority_medium

'Reset the filter back to what the user had selected.
objTaskGrid.SetFilter x, y, z

objTaskGrid.Close
Set objTaskGrid = Nothing

Set objViews = Nothing
```

GetGrid Method

Description Returns a dispatch pointer to a grid object for the Task List view.

Object TaskListView object

Syntax *object*.**GetGrid**

Return type Object/LPDISPATCH

GetLastError S_CON_DOC

Example See GetColumnCount Method (Grid object).

Views object

The Views object allows the SDK to manage the open views in ACT!. The following property and methods apply only to the Views object.

Property

Name	Parameter(s)	Parameter type(s)	Value type	Access
Count Property	None	*	Short Integer	Read Only

Count Property

Description Returns the number of existing views (including any added Explorer views) in the application.

Object Views object

Syntax *object*.**Count**

Value type Short Integer, Read Only

GetLastError S_NO_OPENDB

Methods

Name	Parameter(s)	Parameter type(s)	Return type
Application Method	None	*	Object/LPDISPATCH
ClearError Method	None	*	Void
CloseAll Method	None	*	Long Integer
Create Method	iViewType szName	Short Integer String	Object/LPDISPATCH
CreateBrowserView Method	szCtrlFile	String	Object/LPDISPATCH
CreateBrowserViewFromUrl Method	szCtrlFile szTitle iFlags	String String Short Integer	Object/LPDISPATCH
CreateEx Method	iViewType szName iState	Short Integer String Short Integer	Object/LPDISPATCH
FindExplorerView Method	szName iViewType	String Short Integer	Object/LPDISPATCH
GetActive Method	None	*	Object/LPDISPATCH
GetLastError Method	None	*	Long Integer
GetView Method	iViewType	Short Integer	Object/LPDISPATCH
GetViewEx Method	True|False iViewType	Boolean Short Integer	Object/LPDISPATCH

Application Method

Description Returns a dispatch pointer to the application object containing the Views object.

Object Views object

Syntax *object*.**Application**

Return type Object/LPDISPATCH

Example

```
'The following code can be put into a function that has been passed
'only the Views object - objViews. We assume in the main function
'that the Application object has already been created.

Dim objApp1 as object
'Get a pointer to the Application object.
Set objApp1 = objViews.Application
'Any Application object methods can manipulate the application.
'End the function.
Set objApp1 = Nothing
```

ClearError Method

Requires ACT! 4.0 or later

Description Clears the last error.

Object Views object

Syntax *object*.**ClearError**

Return type Void

See also GetLastError Method

CloseAll Method

Description Closes all open views and the database. This method is the same as the CloseDB method and is equivalent to selecting CLOSE from the File menu in the application. Returns S_ERROR on failure.

Object Views object

Syntax *object*.**CloseAll**

Return type Long Integer

Create Method

Description Creates a view of the specified type. Only one view of a given type can be created. Returns a dispatch pointer to the newly-created view object if successful or NULL if unsuccessful.

Object Views object

Syntax *object*.**Create**(*iViewType*, *szName*)

Parameters *iViewType* A short integer indicating the type of view to be created. The following table lists the values for this parameter.

Value	View type	Value	View type
1	Contacts view	4	Task List view
2	Contact List view	5	Calendar view
3	Groups view		

szName A string representing the name of the view. szName is ignored except if the type is Explorer view. Explorer views are identified by their name. (Currently ignored.)

Return type Object/LPDISPATCH

GetLastError S_INVALID_VWTYPE, S_NO_EMAIL_SYSTEM, S_NO_OPENDB, S_VIEW_EXISTS

See also GetView Method

Activate Method, Show Method in Common properties and methods.

Example

```
'This example creates a Views object.
Dim objApp as object
Dim objViews as object
Dim objTask as object

'Initialize the Application object.
'This starts ACT! if it is not already running.
Set objApp = CreateObject("ACTOLE.APPOBJECT")
```

```
'Open the database.
objApp.OpenFile C:\My Documents\ACT\Database\ACT5demo.dbf

'Create a Views object.
Set objViews = App.Views

'Create the TaskListView object. This brings up the Task List view.
Set objTask = objViews.Create(4, "MyTaskList")

'Close the Task List.
objTask.Close
Set objTask = Nothing
objViews.CloseAll
Set objViews = Nothing
'Close the Application object.
Set objApp = Nothing
```

CreateBrowserView Method

Requires ACT! 5.0.2 or later

Description Creates and opens an Internet browser view by loading the specified control file. Returns a dispatch pointer to the newly-created view object if successful or NULL if unsuccessful.

Note: Assign a View Bar button or menu item to an Internet view to automatically create it when ACT! starts. See Views and Tabs for more information.

Object Views object

Syntax *object*.**CreateBrowserView** (*szCtrlFile*)

Parameters *szCtrlFile* A string specifying the name and optionally the path of the control file that defines the view. See Views and Tabs for information about creating a control file.

Note: Control files with a .CTL extension in the \NetLinks folder are automatically loaded when ACT! starts. If you do not want to load a control file at startup, add it to a different folder such as the ACT! program files folder, or give it a different extension. Then this method will load the control file.

Return type Object/LPDISPATCH

GetLastError S_SERVER_BUSY, S_NO_OPENDB, S_NOT_FOUND

See also CreateBrowserViewFromUrl Method, GetView Method

Activate Method, Show Method in Common properties and methods.

CreateBrowserViewFromUrl Method

Requires ACT! 5.0.2 or later

Description Creates and opens an Internet browser view and loads the specified URL into the view. Returns a dispatch pointer to the newly-created view object if successful or NULL if unsuccessful.

Object Views object

Syntax *object*.**CreateBrowserViewFromUrl** (*szCtrlFile, szTitle, iFlags*)

Parameters *szCtrlFile* A string specifying the starting URL to load into the new Internet browser view.

szTitle A string specifying the title of the Internet browser view. The title appears in the title bar of the ACT! window and in the ACT! Window menu. Specify a null value for this parameter to use the title of the current Web page as the ACT! Window title.

iFlags A short integer specifying the behavior of the Internet browser view. Specify 0 to display the default toolbar or 1 for no default toolbar.

Return type Object/LPDISPATCH

GetLastError S_SERVER_BUSY, S_MEM_ERROR, S_LIST_ERR

See also CreateBrowserView Method, GetView Method

Activate Method, Show Method in Common properties and methods.

CreateEx Method

Requires ACT! 5.0 or later

Description Creates a view of the specified type in a specified window state. Only one view of a given type can be created. Returns a dispatch pointer to the newly-created view object if successful or NULL if unsuccessful.

Object Views object

Syntax *object*.**CreateEx** (*iViewType*, *szName*, *iState*)

Parameters *iViewType* A short integer representing the type of view to be created. The following table lists the values for this parameter.

Value	View type	Value	View type
1	Contacts view	4	Task List view
2	Contact List view	5	Calendar view
3	Groups view		

szName A string representing the name of the view. szName is ignored except if the type is Explorer view. Explorer views are identified by their name. (Currently ignored.)

iState A short integer indicating the state of the window in which the view will be created. The following table lists the values for this parameter.

Value	View State	Value	View State
1	Normal	3	Minimized
2	Maximized	4	Hidden

Return type Object/LPDISPATCH

See also ViewState Property in Common properties and methods

Example

```
'The following sample opens the Contact View in Maximized state
Set objViews = objApp.Views
'Open the ContactView in Maximized state
Set objContactView = objViews.CreateEx(1,"CV" , 2)
```

FindExplorerView Method

Requires ACT! 4.0 or later

Description Returns a dispatch pointer to the Explorer view with the specified name. This method is used only for Explorer views. Returns NULL if a view with the specified name does not exist.

Note: This method requires an Explorer view that was added using the Adding Extensible Views and Tabs to ACT! component of the ACT! SDK.

Object Views object

Syntax *object*.**FindExplorerView** (*szName, iViewType*)

Parameters *szName* A string representing the name of the Explorer view whose dispatch pointer is to be retrieved.

iViewType A short integer representing the type of Explorer view. The following table lists the values for this parameter.

Value	View type	Value	View type
7	Floating view	101	Group tab view
100	Contact tab view		

Return type Object/LPDISPATCH

GetLastError S_NO_OPEN_DB, S-NOT_FOUND, S_SERVER_BUSY, S_VW-NOT-FOUND

See also GetActive Method, GetView Method

GetActive Method

Description Returns a dispatch pointer to the view currently active view object. Returns NULL on error.

Object Views object

Syntax *object*.**GetActive**

Return type Object/LPDISPATCH

GetLastError S_VW_NOT_FOUND

See also Activate Method, Type Property in Common properties and methods.

Example

```
'This example assumes that the coder has only the
'Views object. (There is an active view.)
'The Case statement gets the view type and depending on that the
'user can add more methods. We assume that the particular view
'has been created using the Create function.

Dim objView as object
'Set the objView pointer to the active view.
Set objView = objViews.GetActive

'Find out which window is active for further operations.
Select Case objView.Type
Case 1
'The active view is the Contacts view
'Add ContactView methods
Case 2
'The active view is the Contact List view
'Add ContactListView methods
Case 3
'The active view is the Groups view
'Add GroupView methods
Case 4
'The active window is the Task List view
'Add TaskListView methods
Case 5
'The active window is the Calendar view
'Add CalendarView methods
End Select
'End the function.
Set objView = Nothing
```

GetLastError Method

Description Returns a long integer representing the last error code for the object. For more information, see Error codes.

Object Views object

Syntax *object*.**GetLastError**

Return type Long Integer

See also ClearError Method

GetView Method

Description Returns a dispatch pointer to an existing view of the specified type, except Explorer views. Returns NULL if no view of the specified type exists. If the view does not exist, call the Create method to create the view before calling this method.

Object Views object

Syntax *object*.**GetView** (*iViewType*)

Parameters *iViewType* A short integer representing the type of the view whose dispatch pointer is to be retrieved. The following table lists the values for this parameter.

Value	View type	Value	View type
1	Contacts view	4	Task List view
2	Contact List view	5	Calendar view
3	Groups view		

Return type Object/LPDISPATCH

See also Create Method, FindExplorerView Method

GetLastError S_INVALID_INPUT, S_NO_OPENDB, S_VW_NOT_FOUND

Example

```
'The following code can be put into a function that has been passed
'only the Views Object - objViews. We assume in the main function
'that the Group view and Task view have already been created.

Dim objGrp1 as object
Dim objTask1 as object

'Get the GroupView pointer.
Set objGrp1 = objViews.GetView(3)

'Now any GroupView methods can be applied to manipulate the Group view

'Get the pointer to the TaskListView.
Set objTask1 = objViews.GetView(4)

'TaskListView methods can be applied to manipulate the Task List view.

'End the function.
Set objGrp1 = Nothing
Set objTask1 = Nothing
```

GetViewEx Method

Requires ACT! 5.0.2 or later

Description Returns a dispatch pointer to an existing view of the specified type, except Explorer views, and optionally retain the current state of the view or make it active. Returns NULL if no view of the specified type exists. If the view does not exist, call the Create or CreateEx method to create the view before calling this method.

Note: If you obtain the dispatch pointer of a view silently (preserve the current active or hidden state of the view), calling other methods of the returned dispatch pointer may switch to the view in the ACT! application.

Object Views object

Syntax *object*.**GetViewEx** (*True*|*False, iViewType*)

Parameters *True*|*False* Specify True to retrieve a dispatch pointer to an existing view silently by retaining the current state of the view (active or hidden) or False to display the view and make it the current view.

iViewType A short integer representing the type of the view whose dispatch pointer is to be retrieved. The following table lists the values for this parameter.

Value	View type	Value	View type
1	Contacts view	4	Task List view
2	Contact List view	5	Calendar view
3	Groups view		

Return type Object/LPDISPATCH

See also Create Method, CreateEx Method, FindExplorerView Method, GetView Method

GetLastError S_INVALID_INPUT, S_NO_OPENDB, S_VW_NOT_FOUND

Chapter 6
The Scripting and Command Objects

This chapter discusses the:

- ACT! Scripting Object - Allows for use of event notification.
- Event notification - Permits notification of ACT! events to be used in the SDK.
- ACT! Command object - Provides methods for adding commands to the ACT! graphical interface to run external applications.

Note All objects should be declared and created before use and closed and set to nothing afterwards, implicitly (as in a macro) or explicitly. Using close methods before setting objects to null ensures all memory is released and significantly improves performance. Examples in this document illustrate concepts only, and do not always contain all of these steps.

Scripting object

Third-party applications developed in any language with container support for ActiveX controls (such as VBScript, Visual Basic, and Visual C++) can receive immediate notification of ACT! events and act on them. The ACT! events include opening and closing an ACT! database as well as changing records in the Contact and Group views and the Contact and Group lists.

Scripts can use all objects, methods, and properties in the ACT! Application class; however, the objects, methods, and properties in the Database class are not available for use in scripts. For additional information, see Using scripting support with the Application object.

This section describes the ACT! Scripting Support component of the ACT! Software Development Kit (SDK). This component consists of this set of instructions on how to create scripts that can be notified of events in the ACT! application.

This document assumes that you are familiar with and using the following:

- ACT! 4.0 or later for Windows
- Microsoft Windows 95/98/2000/Me/XP or Windows NT 4.0

System requirements

ACT! scripting support requires:

- ACT! 4.0 or later for Windows
- Microsoft Windows 95/98/2000/Me/XP, or Windows NT 4.0
- Microsoft Internet Explorer version 3.0 or later

Adding a VBScript script file to ACT!

The following steps describe how to add a VBScript file to ACT!

1. Create a text file containing your VBScript code, named with an extension of .VBS, such as MYFILE.VBS. ACT! requires the extension name of .VBS to recognize the file as a VBScript file.
2. Using toolbar customization feature of ACT!, add a new command and specify the name of the VBScript file on the command line. Then add a button to the toolbar and associate the newly created command with that button. If you are not familiar with toolbar customization features of ACT!, see online Help for more information.
3. After you have completed adding the button, it will appear on the ACT! toolbar. Click the added button to execute the script.

Registering the custom control

When ACT! for Windows is installed, the OLE custom control ACTEVENT.OCX is automatically installed and registered in the folder containing the ACT! executable file. If ACTEVENT.OCX is not registered, you can register it by executing ACTREG.EXE. ACTREG.EXE is automatically installed in the folder containing the ACT! executable file.

To get notification of ACT! events in a Visual Basic application

1. Start Visual Basic.
2. Select **CUSTOM CONTROLS** from the **Tools** menu (Visual Basic 4) or **COMPONENTS** from the Project menu (Visual Basic 5).
3. Click **Browse** and select **ACTEVENT.OCX** in the ACT folder.

   The ACT! icon is added to the Visual Basic Toolbox.
4. Drag the ACT! icon onto the Visual Basic Form.

   By default, the name is ActEvent1. The object has no visible properties.

To get notification of ACT! events in a C++ application:

1. Start C++ and create a new MFC project.
2. In the Dialog Editor, select **Insert OLE Control**.
3. In the **Insert OLE Control** dialog box, select **ActEvent Control** and click **OK**.

   A control appears with no visible properties.

To include a notification of an ACT! event:

- In the ClassWizard, select an ACT! event in the Messages scroll list and click Add Function.

Using scripting support with the Application object

Scripts can use all Application class objects, methods, and properties. To use the objects, methods, and properties of the ACT! Application object, include the word "Application" in the script. For example, you can include the following in the script to create the Views object and a Contact view object:

```
objViews = Application.Views
objViews.Create(1,"CV")
```

Note Additional examples of Scripting Support are available in ACT! SDK samples in the Event_Script Samples\Scripts folder.

Following is an example script that shows how the ACT! OLE Application object can be used inside a VBScript added to the ACT! application.

```
Sub ShowMessageBox(ByVal strMsg)
   Msgbox strMsg
End Sub
'Display current ACT! caption to user
ShowMessageBox Application.Caption
'Now change the ACT! application caption
Application.Caption = "My Copy of ACT!"
'Now display the new ACT! caption to the user
ShowMessageBox Application.Caption
```

The following sample script illustrates how to perform calculations on field values. It adds the values in two User fields and places the result in a third User field.

```
'This script demonstrates the calculation of User fields. It adds the values
'in the User1 and User2 fields and places the result in the User3 field.
'It assumes that User1, User2, and User3 are all Currency fields.
'If either the User1 or the User2 field is blank, it is assumed to be 0.
'Use a similar script to perform any other calculation.

Set objViews = Application.Views
Set objContactView = objViews.Create(1, "CV")

objContactView.SetActiveTab "User Fields"
'Assuming that User1, User2 and User 3 are Currency fields
'If User1 or User2 is blank then calculate as if 0
Val1 =objContactView.GetField(50)
If Val1 = "" then Val1 = 0
   Val2 = objContactView.GetField(51)
If Val2 = "" then Val2 = 0
   'Calculate User1 + User2
   Val3 = CCur(Val1) + CCur(Val2)
'Set User3 as the result
objContactView.SetField 52, Val3
```

Scripting methods

Name	Parameter(s)	Parameter Type(s)	Return Type
Register Method	lparam	Long Integer	Long Integer
UnRegister Method	None	*	Long Integer
IsActRunning Method	None	*	Boolean

Register Method

Description Enables the reception of ACT! event notification. This method is typically called at the beginning of the program to enable the reception of messages. ACT! should be running when you call this method.

Object Scripting object

Syntax *object*.**Register** (*lparam*)

Parameters *lparam* A long integer. This parameter is ignored.

Return type Long Integer

Comments This method returns one of the following values:

Value	Meaning
> 0	Any positive integer indicates that ACT! is running.
0	ACT! is not running.
-1	The event is already registered. An event cannot be registered more than once.
<-1	An undefined error occurred.

Example

```
Dim i as long
i = ActEvent1.Register(0)
If i < 1 Then
    If ActEvent1.IsActRunning = False Then
        MsgBox "Please bring up ACT! and start again."
    Else
        MsgBox "Problem registering ACT! Event."
        Please run actreg.exe and try again!"
    End If
End If
```

UnRegister Method

Description Disables the receipt of ACT! event notifications. Call this method when you no longer want to receive notification of ACT! events.

Object Scripting object

Syntax *object*.**UnRegister**

Return type Long Integer

Example See Register Method method.

IsActRunning Method

Description Returns True if ACT! is running and False if ACT! is not running. Call this method to verify that the user has not exited from ACT!

Object Scripting object

Syntax *object*.**IsActRunning**

Return type Boolean

Example See Register Method.

Event notification

Event notification support enables users to get notification of events from the ACT! application. This section contains a list of the supported events. Event notification support is provided through ActiveX controls.

Applications developed in any language with container support for ActiveX controls can receive notification of a variety of ACT! events. An ActiveX control is a software component that incorporates ActiveX technology. For additional information on ActiveX controls, visit http://www.download.com/PC/Activex/ or www.microsoft.com.

In ACT! 5.0 or later, when you delete a lookup of contacts or groups, or multiple contacts from the ContactList view, import contacts or groups into the current database, or perform a sort, event notifications are turned off until the process completes. Then the ContactListChange and GroupListChange events are called, which return updated counts of the contacts and groups.

Note Event notification requires use of the Scripting object. You must call Register to receive notification of ACT! events. Use UnRegister when to end notification and IsActRunning to verify that ACT! is still running.

Event notification events

The following methods are included for setting up event control to begin or end notification of ACT! events or to verify that ACT! is running.The following table lists the events supported.

Name	Called when the user:
OnContactAdd Event	Adds a new contact or saves the record for the current contact. Returns a null string if the user adds a contact or the Unique ID of the current contact when the user saves a new contact.
OnContactChange Event	Makes any change to the record for the current contact. Returns the Unique ID of the current contact.
OnContactDelete Event	Deletes a contact. Returns the Unique ID of the deleted contact.
OnContactListChange Event	Makes any change in the number of contacts in the Contact List. Changes to the number of contacts in the Contact List include changing the lookup, adding a contact, and deleting a contact. Returns an updated count of the contacts in the Contact List.
OnContactLookupChange Event	Makes a change to the current lookup, including adding or deleting a contact or performing a different lookup.
OnContactPosChange Event	Selects a different contact in the Contact view. Returns the Unique ID of the selected contact.
OnDatabaseClose Event	Closes the database.
OnDatabaseOpen Event	Opens a database. Returns the name of the database that is currently open.
OnGroupAdd Event	Adds a new group or saves the current group. Returns a null string if the user adds a group or the Unique ID of the current group when the user saves a new group.
OnGroupChange Event	Makes any change to the current group. Returns the Unique ID of the current group.
OnGroupDelete Event	Deletes a group. Returns the Unique ID of the group that was deleted.
OnGroupListChange Event	Changes the number of contacts in a group (such as adding or deleting a contact) or displays the Group view. Returns an updated count of the contacts in the group.
OnGroupPosChange Event	Selects a different group. Returns the Unique ID of the active group.
OnActUserWantsToClose Event	Selects CLOSE from the File menu or clicks the Close box in the ACT! window.

OnContactAdd Event

Description This event is called when the user adds a new contact or saves the record for the current contact. Returns a null string if the user adds a contact or the Unique ID of the current contact when the user saves a new contact. In ACT! 2000 or later, this event is turned off during the import of contacts into the current database.

Syntax **OnContactAdd**(*szUniqueID*)

Parameters *szUniqueID* A string with either of the following values:

- Null User is adding a new contact.
- Unique ID User has saved a new contact.

Example

```
Private Sub ActEvent1_OnContactAdd(ByVal lpszUniqueID as string)

'Initially when ContactAdd is done, a blank record is generated and
'at that time Unique ID is ""
'After Save the Unique ID gets a value.

If lpszUniqueID <> "" Then
   MsgBox "Contact with Unique ID: " & lpszUniqueID & " added."
End if

End Sub
```

OnContactChange Event

Description This event is called when the user makes any change to the record for the current contact. Returns the Unique ID of the current contact.

Syntax **OnContactChange**(*szUniqueID*)

Parameters *szUniqueID* A string containing the Unique ID of the current contact.

Example

```
Private Sub ActEvent1_OnContactChange(ByVal lpszUniqueID as string)
   MsgBox "Contact with Unique ID: " & lpszUniqueID & "has changed."
End Sub
```

OnContactDelete Event

Description This event is called when the user deletes a contact. Returns the Unique ID of the deleted contact. In ACT! 2000 or later, this event is turned off while deleting a lookup of contacts or while deleting multiple contacts from the Contact List view.

Syntax **OnContactDelete**(*szUniqueID*)

Parameters *szUniqueID* A string containing the Unique ID of the deleted contact.

Example

```
Private Sub ActEvent1_OnContactDelete(ByVal lpszUniqueID as string)
   MsgBox "Contact with Unique ID: " & lpszUniqueID & "deleted."
End Sub
```

OnContactListChange Event

Description This event is called when the user makes any change in the number of contacts in the Contact List. Changes to the number of contacts in the Contact List include changing the lookup, adding a contact, and deleting a contact. Returns an updated count of the contacts in the Contact List.

Syntax **OnContactListChange**(*lCount*)

Parameters *lCount* A long integer indicating the number of contacts now in the Contact List.

Example

```
Private Sub ActEvent1_OnContactListChange(ByVal nCount as long)
   MsgBox "The Contact List has changed. The New Contact Count is" & nCount
End Sub
```

OnContactLookupChange Event

Description This event is called when the user makes a change to the current lookup, including adding or deleting a contact or performing a different lookup. In ACT! 2000 or later, this event is not called while importing contacts into the database, while deleting a lookup of contacts, or while deleting multiple contacts from the Contact List view.

Syntax **OnContactLookupChange**

OnContactPosChange Event

Description This event is called when the user selects a different contact in the Contact view. Returns the Unique ID of the selected contact.

Syntax **OnContactPosChange**(*szUniqueID*)

Parameters *szUniqueID* A string containing the Unique ID of the currently active contact.

Example

```
Private Sub ActEvent1_OnContactPosChange(ByVal lpszUniqueID as string)
   MsgBox "The Current Contact is now: " & lpszUniqueID
End Sub
```

OnDatabaseClose Event

Description This event is called when the user closes the database.

Syntax **OnDatabaseClose**

Example

```
Private Sub ActEvent1_OnDatabaseClose()
   MsgBox "The Database has closed."
End Sub
```

OnDatabaseOpen Event

Description This event is called when the user opens a database. Returns the name of the database that is currently open.

Syntax **OnDatabaseOpen (***szDBName*)

Parameters *szDBName* A string containing the name of the database that is now open.

Example

```
Private Sub ActEvent1_OnDatabaseOpen(ByVal lpszDBName as string)
   MsgBox lpszDBName & " database open."
End Sub
```

OnGroupAdd Event

Description This event is called when the user adds a new group or saves the current group. Returns a null string if the user adds a group or the Unique ID of the current group when the user saves a new group. In ACT! 2000 or later, this event is not called while groups are being imported.

Syntax **OnGroupAdd**(*szUniqueID*)

Parameters *szUniqueID* A string with either of the following values:

- Null User is adding a new contact.
- Unique ID User has saved a new contact.

Example

```
Private Sub ActEvent1_OnGroupAdd(ByVal lpszUniqueID as string)
'Initially when GroupAdd is done, a blank record is generated and
'at that time Unique ID is ""
'After Save the Unique ID gets a value.
If lpszUniqueID <> "" Then
   MsgBox "Group with Unique ID: " & lpszUniqueID & " added."
End Sub
```

OnGroupChange Event

Description This event is called when the user makes any change to the current group. Returns the Unique ID of the current group.

Syntax **OnGroupChange**(*szUniqueID*)

Parameters *szUniqueID* A string containing the Unique ID of the current group.

Examples

```
Private Sub ActEvent1_OnGroupChange(ByVal lpszUniqueID as string)
   If lpszUniqueID <> "" Then
      MsgBox "Group with Unique ID: " & lpszUniqueID & " changed."
   End if
End Sub
```

OnGroupDelete Event

Description This event is called when the user deletes a group. Returns the Unique ID of the group that was deleted. In ACT! 2000 or later, this event is not called during the deletion of a group lookup.

Syntax **OnGroupDelete**(*szUniqueID*)

Parameters *szUniqueID* A string containing the Unique ID of the deleted group.

Example

```
Private Sub ActEvent1_OnGroupDelete(ByVal lpszUniqueID as string)
   If lpszUniqueID <> "" Then
      MsgBox "Group with Unique ID: " & lpszUniqueID & " deleted."
   End if
End Sub
```

OnGroupListChange Event

Description This event is called when the user adds or deletes a group or changes the groups lookup. Returns an updated count of the groups in the Groups view.

Syntax **OnGroupListChange**(*lCount*)

Parameters *lCount* A long integer indicating the number of contacts now in the group.

Example

```
Private Sub ActEvent1_OnGroupListChange(ByVal nCount as long)
   MsgBox "The Group List has changed. The New Group Count is " & nCount
End
```

OnGroupPosChange Event

Description This event is called when the user selects a different group. Returns the Unique ID of the active group.

Syntax **OnGroupPosChange**(*szUniqueID*)

Parameters *szUniqueID* A string containing the Unique ID of the active group.

Example

```
Private Sub ActEvent1_OnGroupPosChange(ByVal lpszUniqueID as string)
   MsgBox "The current group is now: " & lpszUniqueID
End Sub
```

OnActUserWantsToClose Event

Description This event is called when the user selects CLOSE from the File menu or clicks the Close box in the ACT! window.

Syntax **OnActUserWantsToClose**(*bIsFinal*)

Parameter *bIsFinal* A Boolean indicating whether ACT! is closed. A True value indicates that ACT! is closed; a False value indicates that it is not closed.

Comments If ACT! is being controlled by the Application object, when the user closes ACT!, this event is generated twice: Once before the application actually closes (bIsFinal is False) and then again when ACT! closes (bIsFinal isTrue).

Example

```
Private Sub ActEvent1_OnActUserWantsToClose(ByVal bIsFinal As Boolean)
   If bIsFinal = False Then
      'ACT! is given an indication of user wanting to close.
      'It is a good time to utilize the Application object.
      MsgBox "ACT! user wants to close. Uninitializing Application object "
         & vbCrLf & "Closing ACT!"
      'If you are using the Application object, uninitialize it now.
      'It is a good time to uninitialize any other objects you may have
      'references to.
      Set objApp = Nothing
   Else
      ActEvent1.UnRegister
      End
   End If
End Sub
```

Command object

The Command object lets you add commands to the ACT! graphical user interface that run programs outside of ACT!. It provides methods for creating these types of commands, testing whether commands have been created, and adding their associated toolbar buttons and menu items. Also provided are methods for deleting these commands, deleting their associated toolbar buttons and menu items, and determining whether commands exist in toolbars or menus.

The Command object is implemented as an OLE Automation Inproc server. It manages a file named ACTCMD.INI that contains the necessary information to define auxiliary commands that ACT! loads when it starts. The ACTCMD.INI file is created when you run an application using this object. These commands do not appear in the Customize ACT! dialog boxes. The ACTCMD.INI file is created by the ACT! SDK and distributing it to other ACT! users is not recommended. The maximum size of the ACTCMD.INI file is 64 KB.

The object name of the AuxCmds interface is ACTOLE.AUXCMDS. It exports methods that can be used to create, delete, and check for the existence of commands that load external applications. These commands can be added to a toolbar or to the Tools menu for a specific view. This chapter defines these methods.

Adding a command is a two-step process. First, define the command. Then, add the command to a menu or tool bar.

System requirements

To use the OLE Command object, you must have ACT! 5.0 or later.

Error codes

The following error codes apply only to the Command object.

Value	Error code	Description
0	S_OK	Success
102	S_NOT_FOUND	Command or toolbar icon not defined
117	S_INVALID_INPUT	Required input parameter is 0 or null
134	S_MEM_ERROR	Limit of 300 added commands exceeded
157	S_DUPLICATE	Command being added already exists
163	S_INVALID_ID	Invalid View ID specified

Command object methods

The Command object contains information about the custom commands. The following methods apply only to the Command object.

Name	Parameter(s)	ParameterType(s)	Return Type
AddAuxCommand Method	szCommandName szCommandLine szStartIn szToolTip szDescription szSmallIconPath szLargeIconPath iRunState	String String String String String String String Short Integer	Long Integer
AddAuxCommandEnabled Method	szCommandName szCommandLine szStartIn szToolTip szDescription szSmallIconPath szLargeIconPath iRunState	String String String String String String String Short Integer	Long Integer
AddAuxCommandToMenu Method	lViewID szMenuName lPosition lSeparator szCommandName	Long Integer String Long Integer Long Integer String	Long Integer

Name	Parameter(s)	ParameterType(s)	Return Type
AddAuxCommandToToolbar Method	lViewID szCommandName	Long Integer String	Long Integer
AddAuxCommandToToolsMenu Method	lViewID szCommandName	Long Integer String	Long Integer
AddAuxSubMenu Method	lViewID szMenuName lPosition lSeparator szSubMenuName	Long Integer String Long Integer Long Integer String	Long Integer
AuxCommandExists Method	szCommandName	String	Boolean
AuxCommandExistsInMenus Method	lViewID szCommandName	Long Integer String	Boolean
AuxCommandExistsInToolbar Method	lViewID szCommandName	Long Integer, String	Boolean
AuxCommandExistsInToolsMenu Method	lViewID szCommandName	Long Integer String	Boolean
AuxSubMenuExists Method	lViewID szMenuName szSubMenuName lItemCount	Long Integer String String Long Integer	Boolean
DeleteAuxCommand Method	szCommandName	String	Long Integer
RemoveAuxCommandFromMenus Method	lViewID szCommandName	Long Integer String	Long Integer
RemoveAuxCommandFromToolbar Method	lViewID szCommandName	Long Integer String	Long Integer
RemoveAuxCommandFromToolsMenu Method	lViewID szCommandName	Long Integer String	Long Integer
RemoveAuxSubMenu Method	lViewID szMenuName szSubMenuName	Long Integer String String	Long Integer

AddAuxCommand Method

Description Creates an auxiliary command that can be added later to the Tools menu or the standard toolbar of a specified view to begin execution of a program outside of ACT! Use this method before you use the AddAuxCommandToToolsMenu method or the AddAuxCommandToToolbar method. This method returns one of the following: S_OK, S_INVALID_INPUT, or S_DUPLICATE.

Objects Command object

Syntax *object*.**AddAuxCommand** (*szCommandName, szCommandLine, szStartIn, szToolTip, szDescription, szSmallIconPath, szLargeIconPath, iRunState*)

Parameters *szCommandName*String that uniquely identifies the command. The command name must be unique; two commands cannot have the same name.

*szCommandLine*String that specifies the command to be executed.

szStartIn String that specifies the folder containing the data files associated with the command.

szToolTip String that specifies the tool tip text displayed for the command.

szDescription String that describes the command.

*szSmallIconPath*String specifying the path of an icon file that contains a 16 x 16 button image for the command when small buttons are used.

szLargeIconPath String specifying the path of an icon file that contains a 32 x 32 button image for the command when large buttons are used.

iRunState Short integer that specifies the initial state of the window.

The following table lists the value for each state:

Value	State
0	Run the window in the background
1	Normal
2	Maximize the window

Return type Long Integer

Example

```
'This example adds custom command CMDTEST
Dim objCommands As Object
Dim ret As Long

'Create the Commands object
Set objCommands = CreateObject("ACTOLE.AUXCMDS")

'Add custom command Test.exe to run in the normal state
ret = objCommands.AddAuxCommand("CMDTEST", "c:\SDKTest\Test.exe",
      "c:\SDKTest","MyTooltip", "Tests Description", "c\Test\small.ico",
      "c:\Test\large.ico", 1)

If ret <> 0 Then
      List1.AddItem "Error adding custom command "
Else: List1.AddItem "Custom command added successfully"
End If

'Clear the object
Set objCommands = Nothing
```

AddAuxCommandEnabled Method

Requires ACT! 5.0.2 or later

Description Creates an auxiliary command that can be added later to a menu or the standard toolbar of a specified view to begin execution of a program outside of ACT! Use this method before you use the AddAuxCommandToMenu, AddAuxCommandToToolsMenu, or the AddAuxCommandToToolbar method. This method returns one of the following: S_OK, S_INVALID_INPUT, or S_DUPLICATE.

Note: Use this method instead of AddAuxCommand when you need to set the enabled/disabled status of an added command. The AddAuxCommand command always enables an added command. Using AddAuxCommandEnabled, you can set the enabled status of an added command to that of a specified pre-defined ACT! command. For example, the pre-defined Copy command is enabled only if text is selected to copy. You could set the enabled status of a custom command to the status of the Copy command (Command ID 303).

Objects Command object

Syntax *object*.**AddAuxCommandEnabled** (*szCommandName, szCommandLine, szStartIn, szToolTip, szDescription, szSmallIconPath, szLargeIconPath, iRunState, iCommandID*)

Parameters *szCommandName*String that uniquely identifies the command. The command name must be unique; two commands cannot have the same name.

*szCommandLine*String that specifies the command to be executed.

szStartIn String that specifies the folder containing the data files associated with the command.

szToolTip String that specifies the tool tip text displayed for the command.

szDescription String that describes the command.

*szSmallIconPath*String specifying the path of an icon file that contains a 16 x 16 button image for the command when small buttons are used.

*szLargeIconPath*String specifying the path of an icon file that contains a 32 x 32 button image for the command when large buttons are used.

iRunState Short integer that specifies the initial state of the window. The following table lists the value for each state:

Value	State
0	Run the window in the background
1	Normal
2	Maximize the window

iCommandID Short integer that specifies the Command ID of a predefined command whose enabled status is used to set the status of the added command.

Note: Do not use Command ID 104 (File > SAVE) because the Save command is disabled before another command is executed.

For a list of values for this parameter, see Command IDs.

Return type Long Integer

Example

```
'This example adds a command that is linked to Edit > Paste

Dim objAuxCmd As Object
Dim ret As Long
Set objAuxCmd = CreateObject("ACTOLE.AUXCMDS")

szCommandName = "Sample Command"
szCommandLine = "c:\SDKTest\Test.exe"
szStartIn = "c:\SDKTest"
szToolTip = "Tool Tip"
szDescription = "Description"
szSmallIconPath = "small.ico"
szLargeIconPath = "large.ico"
iRunState = 2       'Maximized
iCommandID = 304    'Edit > Paste
ret = objAuxCmd.AddAuxCommandEnabled(szCommandName, szCommandLine,
      szStartIn, szToolTip, szDescription, szSmallIconPath,
  szLargeIconPath,
      iRunState, iCommandID)
```

AddAuxCommandToMenu Method

Requires ACT! 5.0.2 or later

Description Adds the specified command to a menu or a submenu of a specific view. Before using this method you must have added the command using the AddAuxCommand method. Use this method multiple times to insert a group of commands on a menu. This method returns one of the following: S_OK, S_INVALID_ID, S_INVALID_INPUT, S_NOT_FOUND, or S_DUPLICATE.

Objects Command object

Syntax *object.***AddAuxCommandToMenu** (*lViewID, szMenuName, lPosition, lSeparator, szCommandName*)

Parameters *lViewID* Long integer specifying the view in which the specified command will be added to the specified menu.

The following table lists the value for each view:

Value	Target View	Value	Target View
0	Startup View (the view displayed when a database is not open)	4	Task List View
1	Contact View	5	All Calendar Views
2	Contact List View	6	Email View
3	Groups View		

szMenuName String that identifies the menu and optionally the submenu to contain the added command. The value must match the name of a menu in the specified view. Specify a menu name and submenu name in the following format:

```
MenuName\SubMenuName
```

lPosition Long integer specifying the index number of the menu item before which to insert the added command, in the range from 0 to the total number of items in the menu. (Separators are not considered as items.) Specify 0 to insert the command at the top of the menu. Specify the value for the total number of items in the menu to insert the command at the bottom of the menu. You can specify ?1 to always append the command to the bottom of a menu.

lSeparator Long integer specifying the placement of separators around the added command(s).

The following table lists the values for this parameter:

Value	Placement	Value	Placement
0	No separator	3	Top and bottom separators
1	Top separator	-1	Adds separators at the top and bottom of a group of adjacent added commands
2	Bottom separator		

Note: If you are adding a group of commands and submenus with separators at the top and bottom of the group, specify a value of -1 for this parameter when adding each command in the group.

*szCommandName*String that uniquely identifies the command. The command name must be unique; two commands cannot have the same name. A command name can be used only once per view.

Return type Long Integer

Example

```
'This example adds a command to a submenu
Dim objAuxCmd As Object
Dim ret As Long
Set objAuxCmd = CreateObject("ACTOLE.AUXCMDS")

szCommandName = "Sample Command"
If objAuxCmd.AuxCommandExists(szCommandName) Then
      lViewID = 1          'Contact view
      szMenuName = "Edit"
      lPosition = -1       'append submenu to bottom
      lSeparator = 1       'top separator
      szSubMenuName = "Sample Submenu"
      ret = objAuxCmd.AddAuxSubMenu(lViewID, szMenuName, lPosition,
  lSeparator,
         szSubMenuName)
      If objAuxCmd.AuxSubMenuExists(lViewID, szMenuName, szSubMenuName,
         lCmdCount) Then
         ret = objAuxCmd.AddAuxCommandToMenu(lViewID, szMenuName & "\" &
            szSubMenuName, lPosition, lSeparator, szCommandName)
      End If
End If
```

AddAuxCommandToToolbar Method

Description Adds the specified command button to the standard toolbar of a specified view. Before using this method you must have added the command using the AddAuxCommand method. This method returns one of the following: S_OK, S_INVALID_ID, S_INVALID_INPUT, S_NOT_FOUND, or S_DUPLICATE.

Objects Command object

Syntax *object*.**AddAuxCommandToToolbar**(*lViewID, szCommandName*)

Parameters *lViewID* Long integer that specifies the view.

The following table lists the value for each view:

Value	Target View	Value	Target View
0	Startup View (the view displayed when a database is not open)	4	Task List View
1	Contact View	5	All Calendar Views
2	Contact List View	6	Email View
3	Groups View		

szCommandName Unique name that specifies the command to be added. The command name must be unique; two commands cannot have the same name. A command name can be used only once per view.

Return type Long Integer

Example

```
'This example adds a command to the toolbar in the Contact view
Dim objCommands As Object
Dim ret As Long

'Create the Commands object
Set objCommands = CreateObject("ACTOLE.AUXCMDS")
```

```
'Check if the command exists, only then add it to the toolbar
If objCommands.AuxCommandExists("CMDTEST") Then

      ret = objCommands.AddAuxCommandToToolbar(1, "CMDTEST")

      If ret = 0 Then List1.AddItem "Command successfully added to toolbar "
      Else: List1.AddItem "Error adding the command to the toolbar"

Else
      List1.AddItem "The Command does not exist. Please use the method
         AddAuxCommand first!"
End If

Set objCommands = Nothing
```

AddAuxCommandToToolsMenu Method

Description Adds the specified command to the Tools menu of a specific view. Before using this method you must have added the command using the AddAuxCommand method. This method returns one of the following: S_OK, S_INVALID_ID, S_INVALID_INPUT, S_NOT_FOUND, or S_DUPLICATE.

Objects Command object

Syntax *object.***AddAuxCommandToToolsMenu** (*lViewID, szCommandName*)

Parameters *lViewID* Long integer specifying the view in which the specified command will be added to the Tools menu.

The following table lists the value for each view:

Value	Target View	Value	Target View
0	Startup View (the view displayed when a database is not open)	4	Task List View
1	Contact View	5	All Calendar Views
2	Contact List View	6	Email View
3	Groups View		

szCommandName String that uniquely identifies the command. The command name must be unique; two commands cannot have the same name.

Return type Long Integer

Example

```
'This example adds custom command CMDTEST to the Tools menu in the
'Group view

Dim objCommands As Object
Dim ret As Long

'Create the Commands object
Set objCommands = CreateObject("ACTOLE.AUXCMDS")

'Check if the command exists, and only then add it to the Tools menu
If objCommands.AuxCommandExists("CMDTEST") Then

      ret = objCommands.AddAuxCommandToToolsMenu(3, "CMDTEST")
      If ret = 0 Then list1.AddItem "Command successfully added to Tools
 menu!
         Please restart ACT! to see the results"
```

```
        Else: List1.AddItem "Error adding the command to the Tools menu"
    End If
    'Clear the Commands object
    Set objCommands = Nothing
```

AddAuxSubMenu Method

Requires ACT! 5.0.2 or later

Description Adds the specified menu or submenu (cascading menu) to a menu of a specific view. Use this method multiple times to insert a group of submenus on a menu. This method returns one of the following: S_OK, S_INVALID_ID, S_INVALID_INPUT, S_NOT_FOUND, or S_DUPLICATE.

Objects Command object

Syntax *object.***AddAuxSubMenu** (*lViewID, szMenuName, lPosition, lSeparator, szSubMenuName*)

Parameters *lViewID* Long integer specifying the view in which the specified menu or submenu will be added.

The following table lists the value for each view:

Value	Target View	Value	Target View
0	Startup View (the view displayed when a database is not open)	4	Task List View
1	Contact View	5	All Calendar Views
2	Contact List View	6	Email View
3	Groups View		

szMenuName String that identifies the menu to contain the added submenu. The value must match the name of a menu in the specified view. Specify a null value to add a menu to the menu bar.

lPosition

- When adding a menu to the menu bar: Long integer specifying the index number of the menu (left to right) before which to insert the menu, in the range from 0 to the total number of menus. Specify 0 to insert a menu to the left of existing menus. Specify the value for the total number of menus to insert the menu to the right of existing menus. You can specify ?1 to always append the menu to the right of existing menus.
- When adding a submenu to a menu: Long integer specifying the index number of the menu item before which to insert the added submenu, in the range from 0 to the total number of items in the menu. (Separators are not considered as items.) Specify 0 to insert a submenu at the top of the menu. Specify the value for the total number of items in the menu to insert the submenu at the bottom of the menu. You can specify ?1 to always append the submenu to the bottom of a menu.

lSeparator Long integer specifying the placement of separators around the added submenu(s).

The following table lists the values for this parameter:

Value	Placement	Value	Placement
0	No separator	3	Top and bottom separators

Value	Placement	Value	Placement
1	Top separator	-1	Adds separators at the top and bottom of a group of adjacent added submenus
2	Bottom separator		

Note: If you are adding a group of commands and submenus with separators at the top and bottom of the group, specify a value of -1 for this parameter when adding each submenu in the group.

*szSubMenuName*String that uniquely identifies the menu or submenu. The menu or submenu name must be unique; two menus or submenus cannot have the same name.

Return type Long Integer

Example See AddAuxCommandToMenu

AuxCommandExists Method

Description Determines if an auxiliary command with the specified name exists. Returns True (non-zero) if the command exists or False (zero) if the command does not exist.

Objects Command object

Syntax *object.***AuxCommandExists** (*szCommandName*)

Parameters *szCommandName*Unique name of the auxiliary command.

Return type Boolean

AuxCommandExistsInMenus Method

Requires ACT! 5.0.2 or later

Description Determines if a command with the specified name exists in any menu of the specified view. Use this method to verify if a command exists before deleting it. This method returns True (non-zero) is the specified command exists or False (zero) if the command does not exist.

Objects Command object

Syntax *object.***AuxCommandExistsInMenus** (*lViewID, szCommandName*)

Parameters *lViewID* Long integer specifying the view for the specified command.

The following table lists the value for each view:

Value	Target View	Value	Target View
0	Startup View (the view displayed when a database is not open)	4	Task List View
1	Contact View	5	All Calendar Views
2	Contact List View	6	Email View
3	Groups View		

*szCommandName*String that uniquely identifies a command to determine if it exists in any menu of the specified view.

Return type Boolean

Example See AddAuxCommandToMenu Method, RemoveAuxCommandFromMenus Method

AuxCommandExistsInToolbar Method

Description Checks for the existence of the specified command in the toolbar for the specified view. Returns True (non-zero) if the command exists or False (zero) if the command does not exist.

Objects Command object

Syntax *object.***AuxCommandExistsInToolbar**(*lViewID, szCommandName*)

Parameters *lViewID* Long integer indicating the view.

The following table lists the value for each view:

Value	Target View	Value	Target View
0	Startup View (the view displayed when a database is not open)	4	Task List View
1	Contact View	5	All Calendar Views
2	Contact List View	6	Email View
3	Groups View		

szCommandName Unique name that identifies the command to be found.

Return type Boolean

AuxCommandExistsInToolsMenu Method

Description Checks for existence of the specified command in the Tools drop-down menu of the specified view. Returns True (non-zero) if the command exists or False (zero) if the command does not exist.

Objects Command object

Syntax *object.***AuxCommandExistsInToolsMenu** (*lViewID, szCommandName*)

Parameters *lViewID* Long integer specifying the view. The following table lists the value for each view:

Value	Target View	Value	Target View
0	Startup View (the view displayed when a database is not open)	4	Task List View
1	Contact View	5	All Calendar Views
2	Contact List View	6	Email View
3	Groups View		

szCommandName Unique name that specifies the command to search for.

Return type Boolean

AuxSubMenuExists Method

Requires ACT! 5.0.2 or later

Description Returns the number of menu items contained in a menu or submenu. Use this method to verify if a menu or submenu exists and the number of items it contains before deleting it. This method returns True is the specified menu or submenu exists or False if not.

Objects Command object

Syntax *object.***AddSubMenuExists** *(lViewID, szMenuName, szSubMenuName, lItemCount*)

Parameters *lViewID* Long integer specifying the view for the specified menu or submenu. The following table lists the value for each view:

Value	Target View	Value	Target View
0	Startup View (the view displayed when a database is not open)	4	Task List View
1	Contact View	5	All Calendar Views
2	Contact List View	6	Email View
3	Groups View		

szMenuName String that identifies the menu that contains the submenu. The value must match the name of a menu in the specified view. Specify a null value to check for a menu in the menu bar.

*szSubMenuName*String that identifies the menu or submenu to check if it exists and the number of items it contains. The value must match the name of a menu or submenu in the specified menu in the specified view.

lItemCount Reference to a variable that receives a long integer value representing the number of items contained in the specified menu or submenu. A value of 0 is received if the menu or submenu does not exist or if it contains no menu items.

Return type Boolean

Example See AddAuxCommandToMenu, RemoveAuxSubMenu

DeleteAuxCommand Method

Description Deletes an existing auxiliary command and any associated toolbar and menu items. This method returns one of the following: S_OK, S_NOT_FOUND.

Objects Command object

Syntax *object*.**DeleteAuxCommand** (*szCommandName*)

Parameters *szCommandName*Unique name of the auxiliary command to delete.

Return type Long Integer

Example

```
'This example completely deletes the specified custom command
Dim objCommands As Object
Dim ret As Long

'Create the Commands object
Set objCommands = CreateObject("ACTOLE.AUXCMDS")

ret = objCommands.DeleteAuxCommand("CMDTEST")
If ret <>0 Then
     lstDisplay.AddItem sCommandName & " could not be deleted ."
End If

'Clear the Commands object
Set objCommands = Nothing
```

RemoveAuxCommandFromMenus Method

Requires ACT! 5.0.2 or later

Description Removes the specified command from a menu of a specified view. Use AuxCommandExistsInMenus before using this method to verify that the command to be removed exists. This method returns one of the following: S_OK, S_INVALID_ID, S_INVALID_INPUT, or S_NOT_FOUND.

Objects Command object

Syntax *object*.**RemoveAuxCommandFromMenus** (*lViewID, szCommandName*)

Parameters lViewID Long integer specifying the view in which the specified command will be removed. The following table lists the value for each view:

Value	Target View	Value	Target View
0	Startup View (the view displayed when a database is not open)	4	Task List View
1	Contact View	5	All Calendar Views
2	Contact List View	6	Email View
3	Groups View		

szCommandNameString that uniquely identifies the command to remove. The value must match the name of a command in a menu in the specified view.

Return type Long Integer

Example This example removes a command from a menu in the Contact view if it exists

```
Dim objAuxCmd As Object
Dim ret As Long
Set objAuxCmd = CreateObject("ACTOLE.AUXCMDS")

lViewID = 1     'Contact view
szCommandName = "Sample Command"
If objAuxCmd.AuxCommandExistsInMenus(lViewID, szCommandName) Then
     ret = objAuxCmd.RemoveAuxCommandFromMenus(lViewID, szCommandName)
End If
```

RemoveAuxCommandFromToolbar Method

Description Removes the specified command button from the toolbar of a specified view. This method only deletes the specified command from the toolbar. Use the DeleteAuxCommand method to delete the command. This method returns one of the following: S_OK, S_INVALID_ID, S_INVALID_INPUT, or S_NOT_FOUND.

Objects Command object

Syntax *object*.**RemoveAuxCommandFromToolbar** *(lViewID, szCommandName*)

Parameters *lViewID* Long integer specifying the view. The following table lists the value for each view:

Value	Target View	Value	Target View
0	Startup View (the view displayed when a database is not open)	4	Task List View
1	Contact View	5	All Calendar Views
2	Contact List View	6	Email View
3	Groups View		

*szCommandName*Unique name that identifies the command to be removed.

Return type Long Integer

Example

```
'This example removes the CMDTEST command from the toolbar of the
'Contact view

Dim objCommands As Object
Dim ret As Long

'Create the Commands object
Set objCommands = CreateObject("ACTOLE.AUXCMDS")

'If the Command exists in the toolbar, then remove it from the toolbar
If (objCommands.AuxCommandExistsInToolbar(1, "CMDTEST")) Then
     ret = objCommands.RemoveAuxCommandFromToolbar(1, "CMDTEST")

Else
     lstDisplay.AddItem sCommandName & " does not exist in the toolbar in
 the
        view selected, please try again"
End If

'Clear the commands object
Set objCommands = Nothing
```

RemoveAuxCommandFromToolsMenu Method

Description Removes the specified command name from the Tools menu of a specified view. This method only deletes the specified command from the Tools menu. Use the DeleteAuxCommand method to delete the command. This method returns one of the following: S_OK, S_INVALID_ID, S_INVALID_INPUT, or S_NOT_FOUND.

Objects Command object

Syntax *object*.**RemoveAuxCommandFromToolsMenu** *(lViewID, szCommandName*)

Parameters *lViewID* Long integer specifying the view. The following table lists the value for each view:

Value	Target View	Value	Target View
0	Startup View (the view displayed when a database is not open)	4	Task List View
1	Contact View	5	All Calendar Views
2	Contact List View	6	Email View
3	Groups View		

*szCommandName*Unique name that identifies the command to be removed.

Return type Long Integer

Example

```
'This example removes the requested command from the Tools menu of the
'selected view
Dim objCommands As Object
Dim ret As Long

'Create the Commands object
Set objCommands = CreateObject("ACTOLE.AUXCMDS")
```

```
'If the command exists in the toolbar, then remove it from the toolbar
If (objCommands.AuxCommandExistsInToolsMenu(3, "CMDTEST")) Then
     ret = objCommands.RemoveAuxCommandFromToolsMenu(3, "CMDTEST")

Else
     lstDisplay.AddItem sCommandName & " does not exist in the toolbar in
 the
        view selected, please try again"
End If

Set objCommands = Nothing
```

RemoveAuxSubMenu Method

Requires ACT! 5.0.2 or later

Description Removes the specified menu or submenu of a specific view. Use AuxSubMenuExists before using this method to verify the number of menu items in the menu or submenu to be removed. This method returns one of the following: S_OK, S_INVALID_ID, S_INVALID_INPUT, or S_NOT_FOUND.

Objects Command object

Syntax *object.***RemoveAuxSubMenu** *(lViewID, szMenuName, szSubMenuName*)

Parameters *lViewID* Long integer specifying the view in which the specified menu or submenu will be removed. The following table lists the value for each view:

Value	Target View	Value	Target View
0	Startup View (the view displayed when a database is not open)	4	Task List View
1	Contact View	5	All Calendar Views
2	Contact List View	6	Email View
3	Groups View		

szMenuName String that identifies the menu containing the submenu to remove. The value must match the name of a menu in the specified view. Specify a null value to remove a menu in the menu bar.

*szSubMenuName*String that identifies the menu or submenu to remove. The value must match the name of a menu or submenu in the specified menu in the specified view.

Return type Long Integer

Example

```
This example removes a submenu from the Contact view if it exists
Dim objAuxCmd As Object
Dim ret As Long
Set objAuxCmd = CreateObject("ACTOLE.AUXCMDS")

lViewID = 1          'Contact view
szMenuName = "Edit"
szSubMenuName = "Sample Submenu"
If objAuxCmd.AuxSubMenuExists(lViewID, szMenuName, szSubMenuName,
 lCmdCount)
     Then
     ret = objAuxCmd.RemoveAuxSubMenu(lViewID, szMenuName, szSubMenuName)
End If
```

Chapter 7
Views and Tabs

This chapter describes how to add a view or tab displaying an Internet site. The SDK provides three sample control files as well. These instructions are written for programmers who are developing software that will add views, accessible by a Contact or Group tab or View command, to display HTML content.

This document assumes that you are familiar with and using the following:

- ACT! for Windows, version 4.0 or later
- Microsoft Windows 95/98/2000/Me/XP or Windows NT 4.0

To add a new contact or group view, use the ACT! Design Layouts tool.

Overview

You can extend the views and view tabs of ACT! so that users can view an Internet site or document by choosing a button on the View bar, a command from the View menu, or clicking a tab in the Contact or Group view. For example, you can add a command to the View menu that displays a company's Internet site for users to view order entry information, inventory data, or general company information. The Internet site can be specific to a particular contact, or general for a group.

You specify three types of Internet views:

- Floating view
- Contact tab view
- Group tab view

In a floating view, users can browse the site using a navigation toolbar that mimics the operation of a typical Web browser toolbar. In addition, you can add buttons to the toolbar to navigate to URLs of other Internet sites or to display HTML format files.

You specify the type of view, the URL or document to be displayed, View bar buttons, and navigation toolbar buttons in a control file. You can create any number of control files, but each control file can display only one view. After creating the control file, add it to the NetLinks folder (location specified in General Preferences). The default folder location is C:\PROGRAM FILES\ACT\NetLinks. When the user starts ACT!, all files with a .CTL extension in the folder specified for the NetLinks file type are automatically executed.To avoid loading a control file at startup, add it to a different folder such as the ACT! program files folder, or give it a different extension. To load control files in folders other than NetLinks, use the CreateBrowserView Method in the Views object of the Application object.

System requirements

To extend views and view tabs, you must have ACT! 4.0 or later for Windows and Microsoft Internet Explorer version 3.0.2 (Build 1300) or later.

If a Web site referenced in the control file uses Java, there may be a problem with Microsoft Internet Explorer 3.0.2. This is a known problem that has been corrected in Internet Explorer 4.0 or later.

Control files

The control file specifies the type of view and location of the Internet site or document and the function and appearance of the navigation toolbar. It also specifies if the user can open a view from the View bar or a command on the View menu, or a tab in the Contact or Group view.

The control files are executed when ACT! starts. Because a control file for a Floating view adds a button to the View bar and can add a command to the View menu, you should include an explanation of the button and command to your users.

You must create a control file for each Floating view or tab view. The control file includes two required sections and two optional sections:

Control file header is required to specify the version of the control file.

- The View section is required to define the type of view and HTML content to be viewed.
- The Commands section is ignored in ACT! 5.0 or later. This section is optional for Floating views in ACT! 4.0 and ignored for Contact and Group tab views. For floating views, it specifies the navigation button icons and names. Navigation buttons are automatically defined and not needed in the control file in ACT! 5.0 or later.
- The URL section is optional for Floating views and ignored for Contact and Group tab views. It specifies the navigation toolbar buttons that can be used to display additional Internet sites and HTML documents.

Rules

Keep the following rules in mind as you edit a sample control file:

- The control file must contain at least the control file header and the [VIEW], STARTURL, TITLE, and MENU statements. All other sections and statements are optional. The [COMMANDS] section is ignored in ACT! 5.0 or later.
- You can enter the statements within a section in any order.
- You can add spaces between elements in a statement and blank lines between statements to improve the readability of the control file. However, do not include a space in the section headings [VIEW], [COMMANDS], and [URL].
- Separate arguments in a statement with commas.
- Be sure to include quotation marks where shown in the statement formats. Enclose all filenames, URLs, and tooltip text in quotation marks.
- Be sure all URLs are valid; no verification is performed on URLs for validity.
- You can add comments at the end of a statement or on a separate line. Precede each comment with two pound signs (##).
- If you no longer need a view, you can delete its control file, move the control file to a different folder, or change its extension.

Control File Contents

```
ACT! Browser Control File Version 1.00

[VIEW]
TYPE = FLOATBAR | CONTACTTAB | GROUPTAB
## The starting URL when the view is opened
STARTURL = "url"
## The title that appears as the caption for the view or the tab
TITLE = "window_title"
## The menu item name to add to the View bar and View menu
## The MENU statement is optional for ACT! 5.0.2 or later
MENU = "command_name"
## The tooltip argument requires ACT! 4.0 or ACT! 5.0.2 or later
## For other versions of ACT!, specify a string, such as the value used for
 the
## command_name argument of the MENU statement
BUTTON = "tooltip", icon_id, "library"
or
BUTTON = "tooltip", "bitmap.bmp"
PRIORITY = n

[COMMANDS]
## This section is ignored in ACT! 5.0 or later
nav_button = "tooltip", small_icon_id, "library", large_icon_id, "library"
or
nav_button = "tooltip", "small.ico", "large.ico"

[URL]
## The URL buttons to add to the navigation toolbar
"url" = "tooltip", small_icon_id, "library", large_icon_id, "library"
or
"url" = "tooltip", "small.ico", "large.ico"
```

Definitions

Header. The following statement must be the first line of the control file:

```
ACT! Browser Control File Version 1.00
```

View. The View section of the control file specifies:

- The type of view
- The starting URL or document that is displayed when the user launches the view
- The name of the view to be displayed on the title bar of the view or the tab
- The name of the view to be displayed on the View bar and the View menu
- The button to be displayed on the ACT! View bar
- The priority of the view, relative to other views in the NetLinks folder

The View section can include the following statements:

[VIEW] This statement is required and must be the first statement after the control file header.

TYPE = *FLOATBAR* | *CONTACTTAB* | *GROUPTAB* This statement is required to specify the type of view. Include only one of the following arguments:

- FLOATVIEW - Creates a floating view
- CONTACTTAB - Creates a tab in the Contact view
- GROUPTAB - Creates a tab in the Group view

STARTURL = "*url***"** This statement is required. The argument for this statement is:

url The Universal Resource Locator (URL) of the Internet site or the file name of the HTML format document that is displayed when users choose the view from the View bar or View menu (Floating view), or click the View tab (Contact or Group tab view). The URL or file name must be enclosed in quotation marks.

TITLE = "*window_title***"** This statement is required. For Floating views, this statement specifies the text to appear in the title bar of the view. For Contact tab and Group tab views, it specifies the text to appear on the tab. The argument for this statement is:

window_title The title to appear in the title bar of the Floating view or the text to appear on the tab in the Contact or Group tab view. The text must be enclosed in quotation marks.

MENU = "*command_name***"** This statement is optional in ACT! 5.0.2 or later (required in previous versions of ACT!) for Floating views and is not needed and ignored for Contact and Group views.

command_name Determines the name of the Floating view added as a menu item to the View menu. Prior to ACT! 5.0.2, this name is also added under the button for the view on the View bar. The view name must be enclosed in quotation marks.

BUTTON = *"tooltip", "icon_id", "library"* Determines the View bar button, either by using a default image from the ACTRES.DLL or by using a custom .bmp or .ico file. Syntax is:

```
BUTTON = "tooltip", icon_id, "library"
```

- or -

```
BUTTON = "tooltip", "bitmap.bmp"
```

- or -

```
BUTTON = "tooltip", "icon.ico"
```

(ACT! 5.0.2 or later)

This statement is optional for Floating views and is not needed and ignored for Contact and Group views. Use either format of this statement to specify a standard View bar button or custom bitmap file for the view to be displayed on the ACT! View bar. This statement also specifies the text of the tooltip in ACT! 4.0, and the name for the button in ACT! 5.0.2 or later. The arguments for this statement are:

tooltip The text of the tooltip for the View bar button in ACT! 4.0 and the name for the button in ACT! 5.0.2 or later. This argument is ignored in ACT! 5.0 and 5.0.1, but a string is required in all versions of ACT!. The text must be enclosed in quotation marks. Interact Commerce Corporation recommends using the value specified for the command_name argument of the MENU statement for the tooltip.

icon_id An integer specifying the resource ID of the icon to be displayed on the View bar for the view.

The following table lists IDs for the standard View bar button icons supplied in the ACT! 5.0 ACTRES.DLL file:

Image	Icon ID	Image	Icon ID
	24021		24024
	24022		24025
	24023		

library The name of the file containing the icon. The file name must be enclosed in quotation marks. Specify the ACTRES.DLL file in the \ACT folder to use a standard View bar button icon supplied with ACT! 5.0 or later.

bitmap.bmp The name and path of a bitmap file containing the image for a custom button to be displayed on the View bar for the view. The default path is the \ACT folder.

PRIORITY = *n* This statement is optional. It determines the relative position of either the view on the View menu or the tab in the Contact and Group views If you omit this statement, ACT! uses a default priority of 0 (zero), which is the lowest priority. The argument for this statement is:

n An unsigned integer indicating the priority of the view; the higher the value of n, the higher the priority of the view. The highest priority view appears at the top of the list of extensible views in the View menu or in the leftmost extensible view tab in the Contact and Group views.

[COMMANDS] The Commands section specifies the location of the icon for each navigation button to be displayed on the toolbar. The buttons appear on the toolbar in the order in which they are specified in the control file. The Commands section is optional for ACT! 4.0, and ignored in ACT! 5.0. If you omit the Commands section for ACT! 4.0, no navigation toolbar appears when the user displays the view. This section is ignored for Contact and Group views.

Note This procedure only applies to ACT! 4.0. ACT! 5.0 automatically supplies the navigation toolbar buttons that can be defined in this section.

The syntax is:

```
nav_button= "tooltip", small_icon_id, "library", large_icon_id, "library"
```

- or -

```
nav_button = "tooltip", "small.ico", "large.ico"
```

The arguments for this statement are:

nav_button The name of the button to appear on the navigation toolbar. Must be one of the following: HOME, FORWARD, BACK, STOP or REFRESH.

tooltip The text of the tooltip for the button. The text must be enclosed in quotation marks.

small_icon_id An integer specifying the resource ID of the small icons to be displayed on the navigation tool bar. Resource IDs must be enclosed in quotation marks.

large_icon_id An integer specifying the resource ID of the large icon to be displayed on the navigation toolbar. Resource IDs must be enclosed in quotation marks.

library The name of the file containing the icon. The file name must be enclosed in quotation marks.

small.ico The names of the files containing the small and large custom icons to be displayed on the navigation toolbar. The file names must be enclosed in quotation marks.

large.ico displayed on the navigation toolbar. The file names must be enclosed in quotation marks.

[URL] Include this section to display buttons on the navigation toolbar for access to other Internet sites or documents. The Floating view displays a navigation toolbar that contains typical web browser buttons, and can have buttons added to allow users can view other Internet sites or HTML-format documents. These URL buttons appear to the right of the standard navigation buttons, in the order in which they appear in the control file.

A toolbar with navigation buttons automatically defined by ACT! 5.0 and standard toolbar URL buttons (large buttons shown) supplied with ACT! 5.0 and linked to URLs or HTML documents is shown below:

This section is optional for Floating views, and ignored for Contact and Group views.

Use either of the following formats of this statement to specify the location of the icons for each URL button. Include one statement for each button that you want to appear on the toolbar.

```
"url" = "tooltip", small_icon_id, "library", large_icon_id, "library"
```

- or -

```
"url" = "tooltip", "small.ico", "large.ico"
```

The arguments for this statement are:

url The Universal Resource Locator (URL) of the Internet site or the file name of the HTML format document that is displayed when users click the button on the toolbar. The URL or file name must be enclosed in quotation marks.

tooltip The text of the tooltip for the button. The text must be enclosed in quotation marks.

small_icon_id, An integer specifying the resource ID of the small icon to be displayed on the navigation toolbar. Specify a small and large icon from the same set.

large_icon_id An integer specifying the resource ID of the large icon to be displayed on the navigation toolbar. Specify a small and large icon from the same set.

The following table lists IDs for the standard toolbar button icons supplied with ACT! 5.0:

Set	Image	Small icon ID	Image	Large icon ID
1		24001		24011
2		24002		24012
3		24003		24013
4		24004		24014
5		24005		24015

library The name and path of the file containing the icon. The file name must be enclosed in quotation marks. Specify the ACTRES.DLL file in the \ACT folder to use a standard toolbar button icon supplied with ACT! 5.0.

small.ico The name of the file containing the small icon to be displayed on the navigation toolbar. The file names must be enclosed in quotation marks.

large.ico The name of the file containing the small icon to be displayed on the navigation toolbar. The file names must be enclosed in quotation marks.

View bar graphics

For an ACT! 5.0 custom View bar button, create a 32 x 32 pixel bitmap image. ACT! automatically reduces the large image to 16 x 16 for the Mini View bar.

Note For ACT! 4.0, paste a 16 x 16 bitmap image into the upper left corner of the 32 x 32 image.

Navigation toolbar graphics

A set of icons containing matching small (16 x 16) and large (24 x 24) icons is required for each custom button.

To create the small icon file:

1. Create a new icon file.
2. Click the button to the right of the size selector box and select 16 x 16 size.
3. Draw or copy your image onto the new icon.
4. Delete all images except the 16 x 16 image, and then save the new icon file.

To create the large icon file

1. Create a new icon file.
2. Click the button to the right of the size selector box and select custom.
3. Type 24 for both the width and height.
4. Draw or copy your image onto the new icon.
5. Delete all images except the 16 x 16 image, and then save the new icon file.

Using a sample control file

Sample Float, Contact tab, and Group tab view control files are supplied with the ACT! SDK.

To use the sample control file

1. Edit a sample control file using any standard text editor, such as Microsoft Notepad.
2. Rename the sample control file and save it with the .CTL extension, in the folder in the location specified in General Preferences for the NetLinks file type.

   The default folder location is C:\PROGRAM FILES\ACT\NetLinks.
3. Start ACT!.

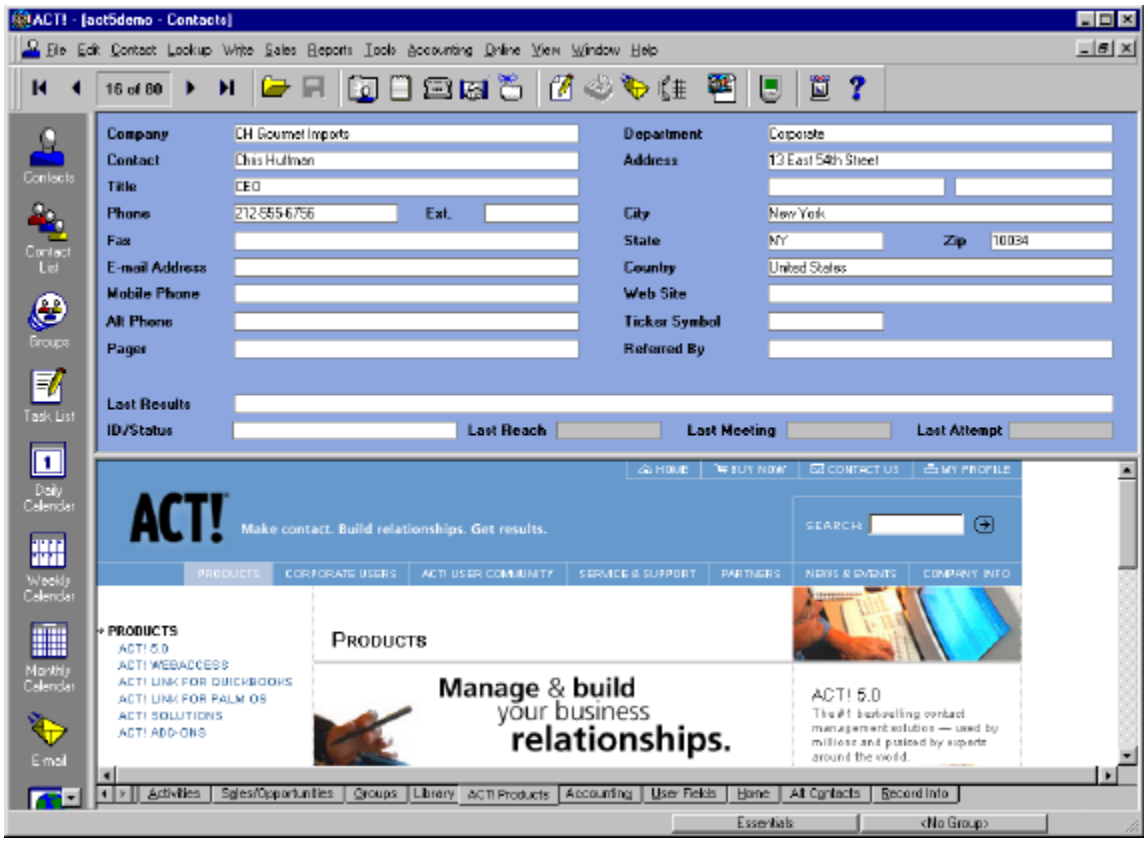
   When started, ACT! executes all files with a .CTL extension in the folder specified for the NetLinks file type.

Note If a control file does not function as expected, check the CONTROL.ERR file in the NetLinks folder for errors such as spelling mistakes, incorrect file names or locations, or other errors. The error file lists the name of the control file, the type of each error detected, and the time each error was generated. Any control file with errors is ignored by the ACT! application.

Samples

Contact tab sample

Contact tab



Control file

```
ACT! Browser Control File Version 1.00
## This is a Contact tab view sample

[VIEW]
TYPE = CONTACTTAB
## The URL to launch when you select the tab
STARTURL = "http://www.act.com/products/index.cfm"
## The title that appears as the caption for the tab
TITLE = "ACT! Products"

PRIORITY = 3
```

Floating view sample

Floating view

Control file

```
ACT! Browser Control File Version 1.00
##This is a Floating view sample

[VIEW]
TYPE = FLOATVIEW

## The starting URL when the view is opened
STARTURL = "http://weather.yahoo.com/"

## The title that appears as the caption for the view
TITLE = "Internet"

## The menu item name that appears on the View bar and the View menu
MENU = "Internet"

## The button for the View bar
BUTTON =  "Internet", 24021, "c:\Program Files\Act\actres.dll"

PRIORITY = 3
```

```
[URL]
## Buttons on the toolbar that link to specified web sites
"http://www.act.com/community/index.cfm" = "User Community", 24003,
    "c:\Program Files\Act\actres.dll", 24013,
    "c:\Program Files\Act\actres.dll"
"http://www.act.com/support/technical support/" = "Technical Support",
 24001,
    "c:\Program Files\Act\actres.dll", 24011,
    "c:\Program Files\Act\actres.dll"
"http://www.actaddons.com/" = "Add Ons", 24002,
    "c:\Program Files\Act\actres.dll", 24012,
    "c:\Program Files\Act\actres.dll"
```

Appendix A
Folder Structure

This appendix describes the various folders that contain ACT! software, templates (for reports, labels, and envelopes), and data (such as contact information, documents, macro definitions, query definitions, and custom layouts).

Folders

Interact Commerce Corporation always recommends backing up any files, especially database and customization files. Follow standard file backup procedures unless otherwise noted.

Act Contains the ACT! application software files. Map files used for importing data into ACT! are stored in the ACT folder with a .MAP extension. Software files in this folder are used internally by ACT! and do not require special backups, but backing up map files is suggested.

BrfCase Stores offline e-mail messages. Files in this folder do not require special backups.

Database folder An ACT! database consists of 20 or more files. For a new installation of ACT! v5.0 (not an upgrade from a prior version of ACT!), the DATABASE folder is placed in the MY DOCUMENTS\ACT folder (PERSONAL\ACT folder in Windows NT 4.0). Windows 2000 conventions specify that applications store user-created data in the MY DOCUMENTS folder by default.

To back up these files, use the Backup command on the ACT! File menu. To include SideACT! data in the backup, ensure the SideACT! Data option on the Options tab of the Backup dialog box is selected.

Drafts folder The DRAFTS folder contains e-mail messages to be sent later. Files in this folder do not require special backups.

Document folder The DOCUMENT folder contains files such as letters, fax cover pages, and memos created in the word-processing application. File extensions for word processing documents are .WPA for the ACT! word processor and .DOC for Microsoft Word. To back up documents, use the Backup command on the ACT! File menu. To include documents in your backup, ensure the Documents option on the Options tab of the Backup dialog box is selected.

For a new installation of ACT! v5.0 (not an upgrade from a prior version of ACT!), the DOCUMENT folder is placed in the MY DOCUMENTS\ACT folder (PERSONAL\ACT folder in Windows NT 4.0). Windows 2000 conventions specify that applications store user-created data in the MY DOCUMENTS folder by default.

Email folder The EMAIL folder contains e-mail messages that are attached to contacts. To back up these messages, use the Backup command on the ACT! File menu. To include e-mail messages attached to contacts, ensure the Attached Mail option on the Options tab of the Backup dialog box is selected.

Layout folder The LAYOUT folder contains default and alternate contact and group layouts, which define how contact and group information is displayed.

Macro folder The MACRO folder contains macros. The file extension for macros is .MPR.

Mail folder The MAIL folder contains subfolders for ACT! Internet mail (if needed). If used, backups are suggested.

NetLinks folder The NETLINKS folder contains text files containing URLs for ACT! Internet links. Back up this folder if you have added custom Internet links in ACT!.

Query folder The QUERY folder contains queries you create and save for later use. The file extension for queries is .QRY.

Report folder The REPORT folder contains templates for creating reports and for creating and printing labels and envelopes. ACT! provides label and envelope templates for use from within a variety of U.S. and international locations.

Spell folder The SPELL folder contains spelling dictionary files. Files in this folder are used internally by ACT! and do not require special backups.

Sync folder The SYNC folder contains database synchronization files. Files in this folder are used internally by ACT! and do not require special backups.

Template folder The Template folder contains ACT! word processor and Microsoft Word templates for documents, including letters, memos, fax cover pages, and forms. You can modify the templates or create your own and save them in this folder.

WPMacros folder The WPMACROS folder contains Microsoft Word macros used by the ACT! application. Files in this folder do not require special backups.

Layouts, reports, and templates

This section lists the default files accompanying the ACT! product.

Documents

Document templates are stored in the \TEMPLATE folder. The following extensions are used for word processor templates: .TPL for ACT! word processor templates and .ADT for Microsoft Word templates. To back up these documents, use the Backup command on the ACT! File menu. To include templates in your backup, ensure the Templates option on the Options tab of the Backup dialog box is selected.

Filename	Description
FAXCOVER.TPL, .ADT	Format for creating a fax cover page using the ACT! word processor or Microsoft Word.
LETTER.TPL, .ADT	Format based on the country specified when you installed ACT! for creating a letter using the ACT! word processor or Microsoft Word.
LTTRELA.TPL, .ADT	Format for creating a letter to send within Europe or Latin America, using the ACT! word processor or Microsoft Word.
LTTRUKA.TPL, .ADT	Format for creating a letter to send within the United Kingdom, Australia, or Asia, using the ACT! word processor or Microsoft Word.
LTTRUSC.TPL, .ADT	Format for creating a letter to send within the United States or Canada, using the ACT! word processor or Microsoft Word.
MEMO.TPL, .ADT	Format for creating a memo using the ACT! word processor or Microsoft Word.

Envelopes

ACT! provides templates for printing envelopes. Modify the envelope templates or create new ones and save them in the \REPORT folder with an extension of .ENV.

To back up these templates, use the Backup command on the ACT! File menu. To include envelope templates, ensure the Envelopes option on the Options tab of the Backup dialog box is selected.

United States and Canada

Filename	Description
6.ENV	3 5/8 x 6 1/2 inches
9.ENV	3 7/8 x 8 7/8 inches
10.ENV	4 1/8 x 9 1/2 inches
11.ENV	4 1/2 x 10 3/8 inches
12.ENV	4 3/4 x 11 inches
MONARCH.ENV	3 7/8 x 7 1/2 inches

Europe, Latin America, United Kingdom, Australia, and Asia

Filename (Europe and Latin America)	Filename (United Kingdom, Australia, and Asia)	Description
C4ELA.ENV	C4UKA.ENV	229 x 324 mm
C5ELA.ENV	C5UKA.ENV	162 x 229 mm
C6ELA.ENV	C6UKA.ENV	114 x 162 mm
C65ELA.ENV	C65UKA.ENV	114 x 229 mm
DLELA.ENV	DLUKA.ENV	110 x 220 mm

Labels

ACT! supplies templates for creating and printing labels. Label templates are based on the standard Avery labels. Modify the label templates or create new ones and save them in the \REPORT folder with an extension of .LBL.

To back up these templates, use the Backup command on the ACT! File menu. To include label templates, ensure the Labels option on the Options tab of the Backup dialog box is selected.

United States and Canada

Labels listed are for laser or ink jet printers, unless dot matrix is specified.

Filename	Description
2160.LBL	Address, mini-sheets, 1 across, 8 down
2162.LBL	Address, mini-sheets, 1 across, 6 down
2163.LBL	Small shipping, mini-sheets, 1 across, 4 down
4014.LBL	Address, dot matrix, 1 across

Filename	Description
4143.LBL	Address, dot matrix, 2 across
4144.LBL	Address, dot matrix, 3 across
4145.LBL	Address, dot matrix, 1 across
4146.LBL	Small shipping, dot matrix, 1 across
4161.LBL	Shipping, red border, dot matrix, 1 across
5160.LBL	Address, 3 across, 10 down
5161.LBL	Address, 2 across, 10 down
5162.LBL	Address, 2 across, 7 down
5163.LBL	Small shipping, 2 across, 5 down
5164.LBL	Shipping, 2 across, 3 down
5385.LBL	Rotary card, 2 across, 4 down
CUSTOM.LBL	Custom label

Europe, Latin America, United Kingdom, Australia, and Asia

Labels listed are for laser or ink jet printers, unless dot matrix is specified

Filename (Europe and Latin America)	Filename (United Kingdom, Australia, and Asia)	Description
CSTMELA.LBL	CSTMUKA.LBL	Custom label
L7159ELA.LBL	L7159UKA.LBL	Address, 3 across, 8 down
L7160ELA.LBL	L7160UKA.LBL	Address, 3 across, 7 down
L7161ELA.LBL	L7161UKA.LBL	Address, 3 across, 6 down
L7162ELA.LBL	L7162UKA.LBL	Address, 2 across, 8 down
L7163ELA.LBL	L7163UKA.LBL	Address, 2 across, 7 down
L7164ELA.LBL	L7164UKA.LBL	Address, 3 across, 4 down
L7165ELA.LBL	L7165UKA.LBL	Parcel, 2 across, 4 down
L7166ELA.LBL	L7166UKA.LBL	Parcel, 2 across, 3 down
L7168ELA.LBL	L7168UKA.LBL	Shipping, 1 across, 2 down
L7169ELA.LBL	L7169UKA.LBL	Parcel, landscape, 2 across, 2 down
L7173ELA.LBL	L7173UKA.LBL	Shipping, 2 across, 5 down

Layouts

ACT! includes 16 contact and 9 group layouts by default. Modify the layouts or create new ones and save them in folder with an extension of .CLY for contact or .GLY for group. Additional layouts may be created, but must be stored in the ACT/Layouts folder. To back up these layouts, use the Backup command on the ACT! File menu. To include layouts, ensure the Layouts option on the Options tab of the Backup dialog box is selected.

Filename	Layout name	Description
640X480.CLY	16-Color Layout	Default contact layout for ACT! v5.0 for systems with a 640 by 480 screen area or a 256 color or less display setting.
640X480.GLY	16-Color Layout	Default group layout for ACT! v5.0 for systems with a 640 by 480 screen area or a 256 color or less display setting.
ACCOUNT5.GLY	Account Layout 2000	Default group layout for ACT! v5.0.
ALTERNAT.CLY	Alternate	Alternate contact layout based on the ACT! 3.0 default layout, but with different borders.
CONTACT1.CLY	Classic Contact 1	Alternate contact layout based on the ACT! 2.0 Contact 1 window.
CONTACT2.CLY	Classic Contact 1	Alternate contact layout based on the ACT! 2.0 Contact 2 window.
DEFAULT.CLY	Contact Layout 3.0	Default contact layout for ACT! 3.0 that can be used as an alternate contact layout in ACT! v5.0.
DEFAULT.GLY	Group Layout 3.0	Default group layout for ACT! 3.0 that can be used as an alternate group layout in ACT! v5.0.
DEFAULT4.CLY	Contact Layout 4.0	Default contact layout for ACT! 4.0 that can be used as an alternate contact layout in ACT! v5.0.
DEFAULT4.GLY	Group Layout 4.0	Default group layout for ACT! 4.0 that can be used as an alternate group layout in ACT! v5.0.
DEFAULT5.CLY	Contact Layout 2000	Default contact layout for ACT! v5.0. An extra copy of this layout is installed in the \ACT folder.
DEFAULT5.GLY	Group Layout 2000	Alternate group layout for ACT! v5.0 that does not display account management fields. An extra copy of this layout is installed in the \ACT folder.
DEFLT16.CLY	Contact Layout (16-Color)	Alternate 16-color contact layout based on the default ACT! 4.0 contact layout.
ESSENTIALS.CLY	Essentials	Default contact layout for ACT! 5.5.
ESSENTIALS2.CLY	Essentials2	Alternate contact layout based on the default ACT! 5.5 contact layout.
ESSENTIALS2.GLY	Essentials2	Default group layout for ACT! 5.5.
ESSENTIALS 800X600.CLY	Essentials800x600	Default contact layout for ACT! 5.5 for systems with a 800 by 600 screen area.
ESSENTIALS 800X600 GRAPHIC.CLY	Essentails800x600graphic	Alternate contact layout for ACT! 5.5 for systems with a 800 by 600 screen area.
LRGFONT.CLY	Large Font	Alternate contact layout based on the ACT! 3.0 default layout, but with larger fonts for easier legibility. This layout can be used as an alternate contact layout in ACT! v5.0.
LRGFONT.GLY	Large Font	Alternate group layout based on the ACT! 3.0 default layout, but with larger fonts for easier legibility. This layout can be used as an alternate group layout in ACT! v5.0.
MODERN.CLY	Modern	Alternate contact layout design for ACT! v5.0.
ROTARY.CLY	Rotary Index	Alternate contact layout design for ACT! v5.0.

Reports

ACT! supplies report templates that allow users to view and print information about contacts, groups, and sales/opportunities. Modify the report templates or create new ones and save them to a file in the \REPORT folder with an extension of .REP. Only templates use the .REP extension; When a user saves a report containing data from the application, ACT! assigns those files the extension .RPT.

To back up report templates, use the Backup command on the ACT! File menu. To include report templates, ensure the Reports option on the Options tab of the Backup dialog box is selected.

Filename	Reports menu command	Report title
ACCCOMP5.REP	Group Reports > Group Comprehensive	Comprehensive Group Report (includes groups and subgroups)
ACCLIST5.REP	Group Reports > Group List	Group List Report (includes groups and subgroups)
ACCMEMB5.REP	Group Reports > Group Membership	Group Membership Report (includes groups and subgroups)
ACCSUMM5.REP	Group Reports > Group Summary	Group Summary Report (includes groups and subgroups)
ACTIVIT5.REP	Activities/Time Spent	Activities / Time Spent
CONTACT5.REP	Contact Report	Contact Report
DIRECTR5.REP	Contact Directory	Contact Directory
GROUP5.REP	Other Report...	Group Report (includes groups only)
GRPLST5.REP	Other Report...	Group List (includes groups only)
GRPMEMB5.REP	Other Report...	Group Membership (includes groups only)
HISTCLA5.REP	History Summary Classic	History Summary
HISTORY5.REP	History Summary	History Summary
HSALLEX5.REP	Other Report...	History Summary Report Example With All Types
NOTEHIS5.REP	Notes/History	Notes/History
PHONELS5.REP	Phone List	Phone List
REFERRA5.REP	Source of Referrals	Source of Referrals
SLSBYMG5.REP	Sales Reports > Sales by Record Manager	Sales by Record Manager
SLSCNTC5.REP	Sales Reports > Sales by Contact	Sales by Contact
SLSDTAI5.REP	Sales Reports > Sales List	Sales List
SLSFRCS5.REP	Sales Reports > Sales Adjusted for Probability	Forecasted Sales Adjusted for Probability
SLSFUNL5.REP	Sales Reports > Sales Pipeline Report	Sales Pipeline Report
SLSTOTA5.REP	Sales Reports > Sales Totals by Status	Sales Totals
STATUS5.REP	Contact Status	Contact Status
TASKLIS5.REP	Task List	Task List

Appendix B
Command IDs

This appendix provides a list of ACT! command IDs, commonly used by the Command Method of the Application class, and the AddAuxCommandEnabled Method of the Command object.

The following values represent Command IDs used in ACT!.

Command ID	Command
101	File > NEW
102	File > OPEN
103	File > CLOSE
104	File > SAVE
105	File > SAVE COPY AS
107	File> PRINT
110	File > PAGE SETUP (Reports/Labels/Envelopes edit mode)
111	File > RUN (Reports/Labels/Envelopes edit mode)
113	Edit > UNDO CHANGES TO CONTACT
120	File > DATA EXCHANGE > IMPORT
121	File > DATA EXCHANGE > EXPORT
124	File > PRINT CURRENT WINDOW
126	Tools > CUSTOMIZE
130	File > SYNCHRONIZE
131	File > SYNCHRONIZE SETUP
140	Edit > DEFINE FIELDS
141	File > DB MAINTENANCE
142	Tools > SCAN FOR DUPLICATES
143	File > DELETE DATABASE
144	File > ADMINISTRATION > DEFINE USERS
145	File > ADMINISTRATION > SET PASSWORD
146	File > ADMINISTRATION > LOCK DATABASE
150	File > EXIT
151	File > SET REMINDERS
160	File > BACKUP
161	File > RESTORE
301	Edit > UNDO
302	Edit > CUT

Command ID	Command
303	Edit > COPY
304	Edit > PASTE
306	Edit > SELECT ALL (Reports/Labels/Envelopes/Design Layouts)
307	Tools > DESIGN LAYOUTS
310	Tools > TIMER
320	Tools > RECORD MACRO
321	Tools > RUN MACRO
322	Tools > DELETE MACRO
325	Edit > REPLACE
326	View > FILTER BY XXX (Grids with filter capabilities and Calendars)
327	Edit > SORT
328	Edit > View > ADD COLUMNS (Grid views)
340	Edit > PREFERENCES
344	Tools > SPELLING
347	Tools > SIDEACT!
349	Tools > PALM HOTSYNCH SETUP
350	Tools > SIDEACT! ALARM SETUP
510	Contact > NEW CONTACT (Contacts view)
511	Contact > DUPLICATE CONTACT (Contacts view)
514	Contact > DELETE CONTACT (Contacts view)
521	Contact > GROUP MEMBERSHIP (Contacts view)
530	Contact > CREATE/EDIT ACTIVITY SERIES
531	Contact > SCHEDULE ACTIVITY SERIES
540	Contact > PHONE CONTACT (Contacts view)
541	Contact > INSERT NOTE (Contacts view) Group > INSERT GROUP NOTE (Groups view)
542	Contact > RECORD HISTORY (Contacts view)
543	Contact > ATTACH FILE (Contacts view) Group > ATTACH FILE (Groups view)
544	Lookup Selected (Contact List view)
545	Omit Selected (Contact List view)
551	Contact > E-MAIL ADDRESSES (Contacts view)
700	Lookup > MY RECORD (Contacts view)
701	Lookup > ALL CONTACTS (Contacts view)
702	Lookup > COMPANY (Contacts view)
703	Lookup > FIRST NAME (Contacts view)
704	Lookup > LAST NAME (Contacts view)
705	Lookup > PHONE (Contacts view)
706	Lookup > CITY (Contacts view)

Command ID	Command
707	Lookup > STATE (Contacts view)
708	Lookup > ZIP CODE (Contacts view)
709	Lookup > ID/STATUS (Contacts view)
710	Lookup > OTHER FIELDS (Contacts view)
711	Lookup > MODIFY MENU
714	Lookup > SYNCHRONIZED RECORDS > DELETED BY REMOTE USERS
715	Lookup > SALES STAGE
716	Lookup > ANNUAL EVENTS
720	Lookup > ALL GROUPS (Groups view)
721	Lookup > OTHER FIELDS (Groups view)
722	Lookup > MODIFY MENU (Groups view)
730	Lookup > PREVIOUS (Contacts view)
731	Lookup > KEYWORD SEARCH (Contacts view)
732	Lookup > BY EXAMPLE (Contacts view)
733	Lookup > INTERNET DIRECTORY (Contacts view)
734	Lookup > E-MAIL ADDRESS
735	Lookup > ACTIVE CONTACTS > MOST ACTIVE
736	Lookup > ACTIVE CONTACTS > LEAST ACTIVE
1301	View > DAILY CALENDAR (Calendar view)
1302	View > WEEKLY CALENDAR (Calendar view)
1303	View > MONTHLY CALENDAR (Calendar view)
1320	Contact > SCHEDULE CALL (Contacts view and Groups view in ACT! 5.0 or later)
1321	Contact > SCHEDULE MEETING (Contacts view)
1322	Contact > SCHEDULE TO-DO CALL (Contacts view and Groups view in ACT! 5.0 or later)
1323	Contact > CLEAR ACTIVITY CALL (Contacts view and Groups view in ACT! 5.0 or later)
1324	Contact > CLEAR MULTIPLE ACTIVITIES CALL (Contacts view and Groups view in ACT! 5.0 or later)
1334	View > FILTER ACTIVITIES (Contacts view)
1337	Contact > RESCHEDULE ACTIVITY (Contacts view and Groups view in ACT! 5.0 or later)
1343	Contact > SEND ACTIVITY (Contacts view and Groups view in ACT! 5.0 or later)
1345	Tools > OUTLOOK ACTIVITIES > UPDATE
1346	Tools > oUTLOOK ACTIVITIES > REMOVE ALL ACTIVITIES
1900	Group > NEW GROUP (Groups view)
1901	Group > DUPLICATE GROUP (Groups view)
1903	Group > DELETE GROUP (Groups view)
1904	Group > GROUP MEMBERSHIP (Groups view)
1914	Group > GROUP MEMBERSHIP > RUN RULES (Groups view) (ACT! 5.0 or later)
1915	Group > NEW SUBGROUP (Groups view) (ACT! 5.0 or later)
1916	Group > MOVE GROUP (Groups view) (ACT! 5.0 or later)

Command ID	Command
1917	Group > GROUP MEMBERSHIP > DEFINE RULES (Groups view) (ACT! 5.0 or later)
1918	Group > GROUP MEMBERSHIP > VIEW RULES (Groups view) (ACT! 5.0 or later)
1919	Group > CREATE LOOKUP
2100	Query > CHECK QUERY STATUS
2101	Query > RUN
2102	Query > CLEAR
2105	Query > SHOW QUERY HELPER
2106	Query > SPECIFY QUERY SORT
2107	Query > CONVERT TO ADVANCED QUERY
2300	Write > LETTER
2301	Write > MEMORANDUM
2302	Write > FAX COVER PAGE
2303	Write > MAIL MERGE
2304	Write > OTHER DOCUMENT
2305	Write > EDIT DOCUMENT TEMPLATE
2306	Write > E-MAIL MESSAGE
2325	Reports > OTHER REPORT
2326	Reports > EDIT REPORT
2327	Reports > MODIFY MENU
2400	Reports > SWAP FIELDS (Edit replace view)
2401	Reports > COPY FIELD (Edit replace view)
2402	Reports > Run (Edit replace view)
2500	View > CONTACTS
2503	View > E-MAIL
2504	View > GROUPS
2505	View > MINI-CALENDAR
2506	View > CONTACT LIST
2507	View > TASK LIST
2509	View > ACTIVITIES TAB
2510	View > NOTES/HISTORY TAB
2513	Edit Mode (Contact List view)
2514	Tag Mode (Contact List view)
2515	View > TAG ALL CONTACTS (Contact List view - Tag mode)
2516	View > UNTAG ALL CONTACTS (Contact List view - Tag mode)
2517	View > GROUPS TAB (Contacts view)
2518	View > CONTACTS TAB (Groups view)
2519	View > SALES/OPPORTUNITIES TAB (ACT! 5.0 or later)
2530	View > INTERNET SERVICES

Command ID	Command
2702	View > FILTER NOTES/HISTORY
2800	Reports > CONTACT REPORT
2801	Reports > CONTACT DIRECTORY
2802	Reports > PHONE LIST
2803	Reports > TASK LIST
2804	Reports > NOTES/HISTORY
2805	Reports > HISTORY SUMMARY
2806	Reports > ACTIVITIES/TIME SPENT
2807	Reports > CONTACT STATUS
2808	Reports > SOURCE OF REFERRALS
2809	Reports > GROUP MEMBERSHIP (ACT! 4.0 or earlier only) Reports > GROUP REPORTS > GROUP MEMBERSHIP (ACT! 5.0 or later)
2810	Reports > MODIFY MENU
2812	Reports > HISTORY SUMMARY CLASSIC
2813	Reports > SALES REPORTS > SALES LIST (ACT! 5.0 or later)
2814	Sales > SALES GRAPH (ACT! 5.0 or later) Reports > SALES REPORTS > SALES GRAPH (ACT! 5.0 or later)
2815	Sales > SALES PIPELINE (ACT! 5.0 or later) Reports > SALES REPORTS > SALES PIPELINE (ACT! 5.0 or later)
2816	Reports > SALES REPORTS > SALES PIPELINE REPORT (ACT! 5.0 or later)
2817	Reports > GROUP REPORTS > GROUP SUMMARY (ACT! 5.0 or later)
2818	Reports > GROUP REPORTS > GROUP COMPREHENSIVE (ACT! 5.0 or later)
2825	Reports > SALES REPORTS > SALES TOTALS BY STATUS (ACT! 5.0 or later)
2826	Reports > SALES REPORTS > SALES ADJUSTED FOR PROBABILITY (ACT! 5.0 or later)
2827	Reports > SALES REPORTS > SALES BY RECORD MANAGER (ACT! 5.0 or later)
2828	Reports > SALES REPORTS > SALES BY CONTACT (ACT! 5.0 or later)
2829	Reports > GROUP REPORTS > GROUP LIST
3036	Objects > ADD LABEL (Report editor)
3037	View > RULER SETTINGS (Report editor)
3038	View > SHOW/HIDE RULER (Report editor)
3039	View > SHOW/HIDE GRID (Report editor)
3040	View > SNAP TO GRID (Report editor)
3045	Objects > MAKE SAME WIDTH (Report editor)
3046	Objects > MAKE SAME HEIGHT (Report editor)
3047	View > SHOW SECTION TITLES (Report editor)
3048	View > SHOW/HIDE TOOL PALETTE (Report editor)
3049	Edit > DEFINE SECTIONS (Report editor)
3050	Edit > DEFINE FILTERS (Report editor)
3300	Sales > NEW SALES OPPORTUNITY (ACT! 5.0 or later)
3301	Sales > VIEW/EDIT SALE (ACT! 5.0 or later)

Command ID	Command
3302	Sales > COMPLETE SALE (ACT! 5.0 or later)
3303	Sales > DELETE SALE (ACT! 5.0 or later)
3307	Sales > MODIFY SALESSTAGES (ACT! 5.0 or later)
4000	Edit > FORMAT (Labels and Envelopes)
5000	Home (Internet Services)
5001	Forward (Internet Services)
5002	Back (Internet Services)
5004	Stop (Internet Services)
5005	Refresh (Internet Services)
20000	Window > CASCADE
20001	Window > TILE HORIZONTAL
20002	Window > TILE VERTICAL
20206	Help > HOW TO USE HELP
20209	Help > ACT! UPDATE
20251	Contact > VIEW/EDIT ACTIVITY DETAILS (ACT! 5.0 or later)

Index

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W