GroupWise® Software Developer Kit

Object API

November 2012

Novell

Legal Notices

Novell, Inc. makes no representations or warranties with respect to the contents or use of this documentation, and specifically disclaims any express or implied warranties of merchantability or fitness for any particular purpose. Further, Novell, Inc. reserves the right to revise this publication and to make changes to its content, at any time, without obligation to notify any person or entity of such revisions or changes.

Further, Novell, Inc. makes no representations or warranties with respect to any software, and specifically disclaims any express or implied warranties of merchantability or fitness for any particular purpose. Further, Novell, Inc. reserves the right to make changes to any and all parts of Novell software, at any time, without any obligation to notify any person or entity of such changes.

Any products or technical information provided under this Agreement may be subject to U.S. export controls and the trade laws of other countries. You agree to comply with all export control regulations and to obtain any required licenses or classification to export, re-export, or import deliverables. You agree not to export or re-export to entities on the current U.S. export exclusion lists or to any embargoed or terrorist countries as specified in the U.S. export laws. You agree to not use deliverables for prohibited nuclear, missile, or chemical biological weaponry end uses. Please refer to http://www.novell.com/info/exports/ (http://www.novell.com/info/exports/) for more information on exporting Novell software. Novell assumes no responsibility for your failure to obtain any necessary export approvals.

Novell, Inc.
1800 South Novell Place
Provo, UT 84606
U.S.A.
www.novell.com

Online Documentation: To access the online documentation for this and other Novell developer products, and to get updates, see the Novell Developer Web site (http://www.novell.com/developer). To access online documentation for Novell products, see the Novell Documentation Web site (http://www.novell.com/documentation).

Novell Trademarks

For Novell trademarks, see the Novell Trademark and Service Mark list (http://www.novell.com/company/legal/trademarks/tmlist.html).

Third-Party Materials

All third-party trademarks are the property of their respective owners.

Contents

About This Guide

GroupWise Object API lets you see, use, and manipulate the GroupWise information store from outside GroupWise. Object API lets you use the GroupWise Address Book and document management capabilities to affect mail messages, appointments, tasks, notes, and phone messages.

IMPORTANT: Unless otherwise marked, the features in GroupWise Object API works with GroupWise 8 and later versions.

This guide contains the following sections:

Audience

This guide is intended for GroupWise developers.

Feedback

We want to hear your comments and suggestions about this manual and the other documentation included with this product. Please use the User Comment feature at the bottom of each page of the online documentation, or go to Novell Documentation Feedback (http://www.novell.com/documentation/feedback.html) and enter your comments there.

Additional Information

For additional GroupWise SDK documentation, see the Novell Developer Web site (http://www.novell.com/developer).

1 Overview

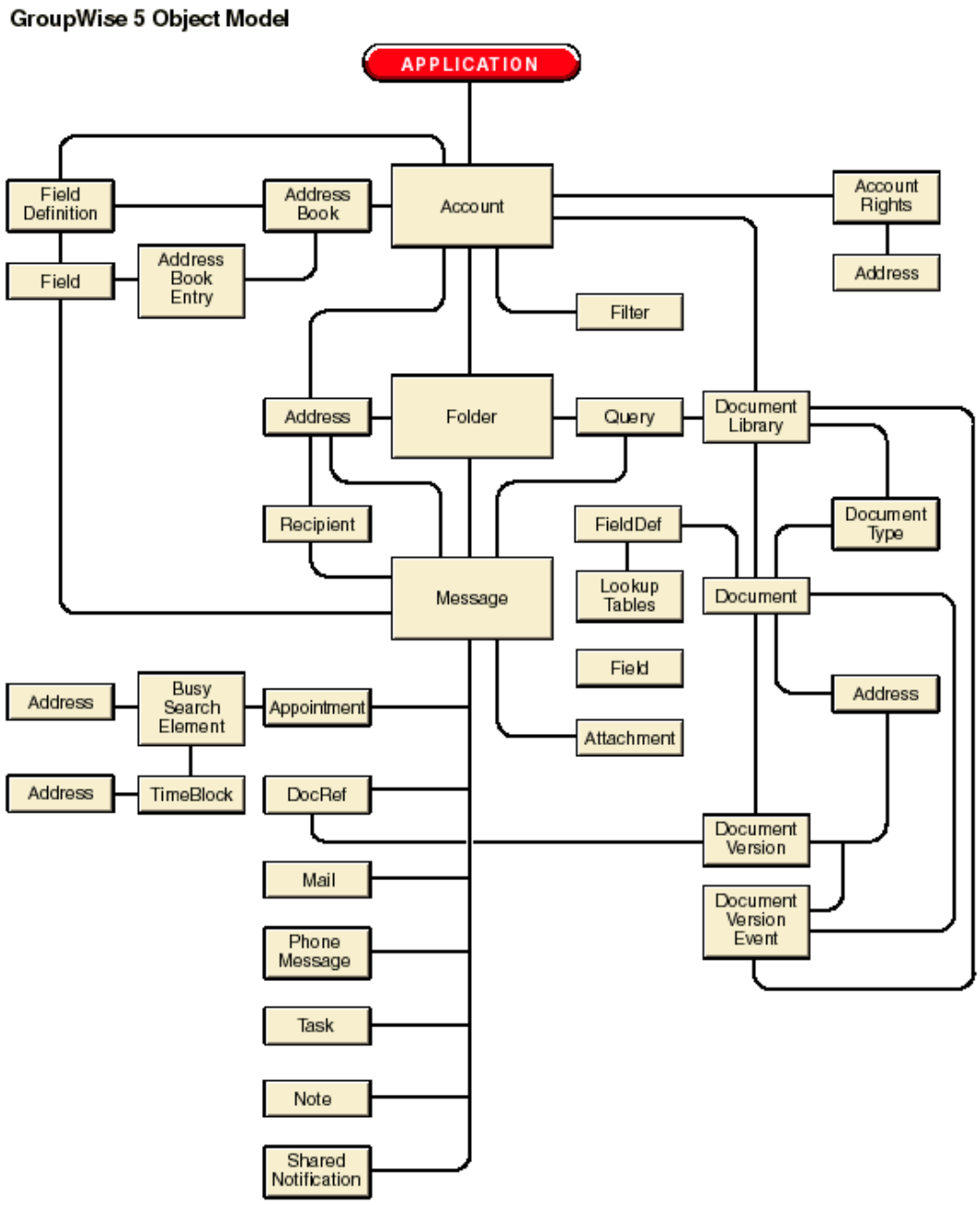
You can use GroupWise Object API through OLE languages such as Visual Basic and Delphi, and object-oriented languages such as C++. Object API supports OLE Automation, which is an industry standard for interfacing applications.

IMPORTANT: To synchronize with the master store, you need the GroupWise client (along with the GWSync.exe file). If you run the Object API against the cache without running the client, the changes will not be synchronized to the master store until you run the client.

This section discusses the basic concepts to understanding how to use the GroupWise Object APIs to extend the GroupWise client. It discusses:

GroupWise Object API defines the objects shown in the following relational object diagram:

Figure 1-1 *GroupWise Object Model*



1.1 User Interface

The Object API does not define a user interface other than a login dialog box. An application with its own user interface can use the Object API to access the GroupWise information store.

Third-party applications that run under the GroupWise Client can use a Custom 3rd-Party Object (C3PO). The C3PO interface simplifies extending GroupWise functionality. The C3PO specification replaces custom messages and custom commands in the GroupWise Client. Using the Object API, C3POs access and manipulate the GroupWise information store. C3POs must be registered in the Windows 95 system registry but the Object API does not use the system registry.

1.2 Object-Oriented Design

GroupWise Object API design is based on state-of-the-art object-oriented techniques and theory. Object API objects provide an organized representation of the GroupWise information store. An object contains the relevant properties (data) and the methods (functions) which manipulate the properties.

New objects may be defined without affecting older applications that are not aware of the new objects. Additional properties and methods may be added to the existing objects without changing the previous functionality of the object. Thus, new versions of the Object API are inherently backward compatible.

Any development tools that can access OLE Automation objects can be used to develop applications and obtain GroupWise information.

OLE Automation can manage collection objects defined in the Object API. A collection object is a special object that provides a uniform means of managing a number of objects of a given type.

1.3 Object API versus Other APIs

The GroupWise Object API does not replace any other API. Other APIs can be used in conjunction with the Object API. GroupWise also supports the Microsoft Messaging API (MAPI), Dynamic Data Exchange (DDE), Open Document Management API (ODMA), and the Third-Party Handler.

MAPI is a standard set of messaging functions used to create mail-enabled applications. However, since MAPI focuses on email, it is not general enough to handle other GroupWise functions such as appointments, tasks, and calendars. An application might use the Object API to access the GroupWise information store and use MAPI to interface the application to another email system.

GroupWise can act as a DDE server and will execute tokens or commands. GroupWise will support ODMA, which is a standardized interface that fits between applications and document management systems.

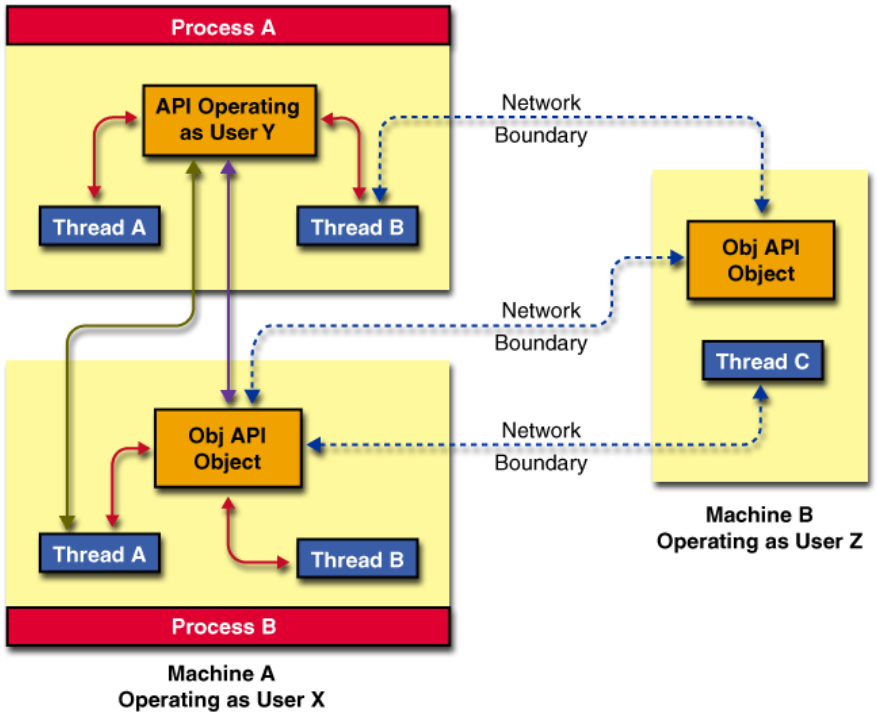
GroupWise also supports the Third-Party Handler, which provides the ability to trap and process tokens that are generated by GroupWise. However, the Object API will provide the information and functionality to interface with GroupWise 8 and later versions and to implement many applications.

More than any other API, the GroupWise Object API opens the GroupWise system to third-party development. The GroupWise Object API is easier to use than APIs such as MAPI because it provides a higher-level interface with GroupWise. It provides access to the GroupWise information store free from the complexity of the underlying system and allows you to focus on using the information encapsulated in the objects.

1.4 Development Path

The Object API is designed to support a full object distribution model. GroupWise does not support this entire model. However, the Object API is designed to survive or adapt into future deployment topologies. This diagram shows an example of a fully distributed model.

Figure 1-2 *Fully Distributed Object Model*



In particular, the API should be capable of multiple objects in different processes to interchange their COM interfaces. For example, a process might pass a pointer to an Account object to another machine in order to proxy to data that is only available on the source machine.

The API is completely compliant with standard COM marshalling techniques, and the interfaces to the objects are capable of accepting a marshalled object.

1.5 Support and Limitations

The following items are supported in GroupWise:

- Proper data typing at the API interfaces to support future distributed objects.
- Complete remoting. The API can be remoted between machines using standard OLE distribution techniques.
- Partial remoting. The API can be used simultaneously on a local machine and on any number of remote machines subject to API locality. That is, object pointers obtained from one process space may not be interchanged with an Object API function in another process space.
- Single model threading. Objects are used on the single thread.
- Apartment threading model. An object can be used only on the thread that created it.

The following items are not supported in GroupWise and must be avoided by the API client:

- Mix-and-Match threading. This is not supported by GroupWise until it is supported by the Win32 COM implementation.

- Simultaneous root user accounts on the same machine. The internal architecture is laid, but the full support will not be surfaced until all component models are worked out.
- Passing of an Object API object, obtained in process A, as a parameter to an Object API method in process space B.

1.6 Windows NT Services

Object API can be run as part of an NT service, with the following exceptions:

1. Call the MultiLogin method instead of the normal Login (page 211) method to access accounts. An NT service cannot display any User Interface (dialogs, etc.) that the normal Login method can display (to prompt for password or in case of a login error). MultiLogin does not display any dialog boxes. It simply returns login errors to the caller.
2. If you must call the Login method, make sure the login will be successful without any other user interaction. An NT service cannot display User Interface. If the Login method has errors or needs a password, it will try to display Dialog Boxes, which violates this rule.
3. Any calls to MultiLogin or Login must pass in the "/ntservice" parameter as part of the login command line. This is the only way that Object API knows it is running as part of an NT service. GroupWise needs this parameter to properly login to the MAPI address book providers.

1.7 GroupWise 6.5 Features

The following parser statements are available in GroupWise 6.5 and later releases:

1.7.1 Change Password Dialog Box

For GroupWise 6.5 and later releases, a GroupWise Administrator can turn off the ability for GroupWise to change an LDAP password. If this occurs, any attempt to call Account::SetPassword returns an error.

If the LDAP authentication is turned on AND their LDAP password has expired, the Object API normally prompts users to change their passwords. If the GroupWise Administrator has set this flag, the normal Change Password dialog appears but all text boxes are disabled, the OK button is disabled, and a new error message is displayed that asks the user to change his LDAP password using other means.

1.7.2 Sent Items Folder

For GroupWise 6.5 and later releases, the Outbox folder was renamed to be the Sent Items folder, which causes a conflict with the current Sent Items query folder. The first time a user runs the GroupWise 6.5 code, it hides the Sent Items query folder or renames it to Sent Items Old.

If the query folder is hidden, GroupWise 6.5 displays only the new Sent Items systems folder. If the query folder is renamed, GroupWise 6.5 displays both the query folder and the new Sent Items system folder while older GroupWise Windows clients continue to show the Sent Items query folder and hide the new Sent Items system folder.

The Object API always allows access to both the query folder and the Sent Items system folder. If the two folders remain with the same name, you must use the ObjType property of the Folder (page 101) object to distinguish between the two folders.

1.7.3 Parser Text Statement

Two new filter statements are available in GroupWise GroupWise 6.5 and later releases. These statements should be added (after the SUBJECT, PLACE, and other text) to the set of available statements for text searches.

```
CATEGORY MATCHES "Sample"
```

This statement returns any item that contains a Category of name Sample.

NOTE: Only the text operation MATCHES works with the Filter expression CATEGORY. All other text operations return an error.

```
PRIMARY_CATEGORY
```

This statement returns any item that contains a Category of name Sample, and for which the Sample Category was marked as the Primary Category for the item.

NOTE: Only the text operation MATCHES works with the Filter expression PRIMARY_CATEGORY. All other text operations return an error.

1.7.4 Parser Unary Statement

A new filter statement is available in GroupWise GroupWise 6.5 and later releases. The statement should be added (after the ON_CALENDAR, OFFICIAL_DOCUMENT_VERSIONS_ONLY, and other text) to the third set of available statements.

```
ON_CHECKLIST
```

This statement returns any item that has been placed on the master Checklist. If a NOT is placed before the statement, the search finds items that are not on the Checklist only.

1.7.5 Parser Numeric Statement

Two new filter statements are available in GroupWise GroupWise 6.5 and later versions and should be added (as part of the CURRENT_VERSION_NUMBER, NUMBER_READ, and other text) to the set of available statements.

TASK_PRIORITY

This keyword returns any item with a Task Priority value that matches the Numeric operation. For example, a task that displays in the GroupWise client with Task Priority A3 can be found with the following filter statements:

```
(TASK_PRIORITY = 3)
 (TASK_PRIORITY > 2)
```

TASK_CATEGORY

This keyword returns any item with a Task Category value that matches the Numeric operation. For example, a task that displays in the GroupWise client with Task Priority "A3" can be found with the following filter statements:

```
(TASK_CATEGORY = "A")
   (TASK_CATEGORY < "B")
```

NOTE: This is a Text value sent in to a numeric operation and follows the pattern of the GroupWise client, which also searches for the Task Priority with the numeric operators =, !, >, <, etc.

The GroupWise client always displays the Task Priority as a numeric value after the Task Category letter value. The Object API, however, split the Task Priority and Task Category into two separate properties. When building a Filter or Query expression, the GroupWise Windows Client splits the shown Task Priority field into the Task Category and Task Priority fields to search. These changes to the Object API follow the GroupWise Client split for separating the two values for a Task search.

1.7.6 GroupWise Error Messages

A detailed list of GroupWise error messages that appear in ConsoleOneTM, GroupWise agents, and GroupWise clients, along with possible causes and suggested actions to resolve the problem, can be found in the Troubleshooting section of the GroupWise 6.5 (http://www.novell.com/documentation/gw65/gw65_tsh1/data/a4ehiyt.html) documentation.

1.7.7 Date Fields

For any Date field that is viewed through the Object API (including the Birthday field), note the following restrictions:

1. The user must be logged in using MultiLogin.
2. The MultiLoginAddressBookSupport (part of the Application4 object) must be set to TRUE.

If these two restrictions are not followed, Object API can display only the String fields as passed by the MAPI Address Book providers.

1.8 Visual Basic

Visual Basic (VB) uses a packager that generates a setup program for applications. However, GroupWise Object API (a GroupWare type library) cannot be installed in a Visual Basic package and needs to be installed separately.

If you try to install Object API with a VB packager, it:

- Finds only the Object API DLL (gwcma1.dll), not the files that gwcma1.dll is dependent on.
- Installs gwcma1.dll in the wrong directory and won't install the dependent files.

1.9 Rules Support

Rules support is available in the Object API with GroupWise version 8.0.0. For a basic understanding of Rules, refer to the rules section of the GroupWise Windows Client User Guide (http://www.novell.com/documentation/gw8/index.html).

For additional information on programmatically handling rules, refer to the GroupWise Web Service (SOAP) documentation (http://developer.novell.com/wiki/index.php/GroupWise_Web_Service_(SOAP)).

2 Tasks

To get started using the Object API in a third-party application, you first need to get the GroupWise Application object and log in. This section provides instructions for the following languages:

2.1 Visual Basic

This section describes how to accomplish the following:

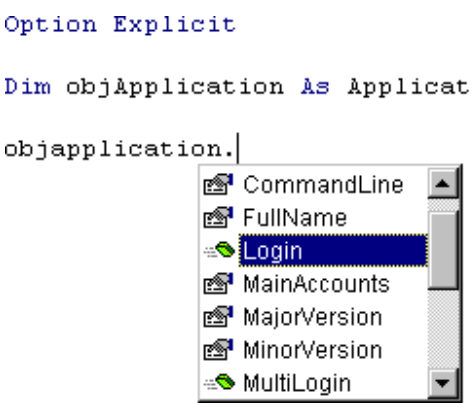
2.1.1 Retrieving Application Objects

In Visual Basic, to get a GroupWise Application object, you can choose between:

Early Binding

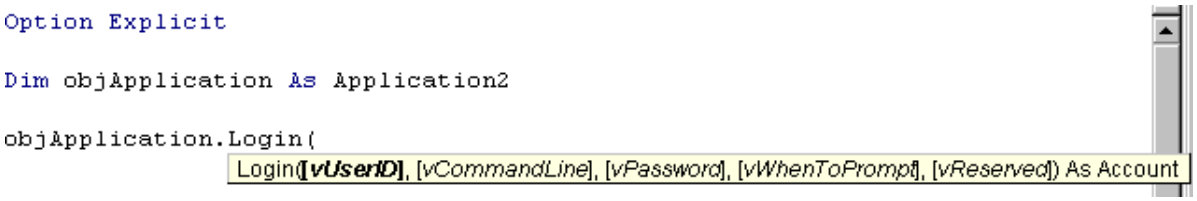
Early Binding is recommended for most uses. It provides advantages in development, debugging, and run-time. The greatest advantage is that it gives your executable faster performance since it doesn't have to look up the CLSIDs (class identifiers) from the Windows registry at run-time. You can declare objects as an application, message, or appointment object, rather than just declaring it as a generic object. After you have declared the objects, the name completion function (shown below) works to complete object, parameter, and method types. It will also provide you the information you need about objects, methods, or parameters.

Figure 2-1 *Name Completion Function*



For example, as you start typing a method, it shows you the parameters you can utilize, as shown below.

Figure 2-2 *Method Popup That Shows Parameters*

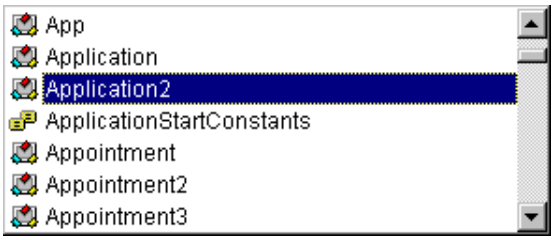


To use early binding, follow these steps:

1. From the Project menu, select References.
2. A dialog appears that lists a group of available references (or libraries). Check the GroupWare type library box and click OK.
3. If you want to view the objects, properties, and methods from the View menu, select Object Browser.

NOTE: Always use the latest version of the objects. For example, if the object has a number next to it, use the last one, such as Application2 or Appointment3 (as shown below).

Figure 2-3 *Choosing Object Versions*



The following illustrates the creation of an instance of the GroupWise Application object with early binding:

```
Set objGroupWise = New Application2
```

Late Binding

Late binding is not recommended for most cases. It provides advantages when you want to get back multiple objects. The disadvantage is that it requires the executable to look up the CLSID from the Windows registry at run-time, making it slower. Late binding uses only the types given by Visual Basic. Thus, you have to declare all objects to be of the generic "Object" or Variant type.

To use late binding, you need to use the CreateObject call to create an generic object.

The following illustrates the creation of an instance of the GroupWise Application object with late binding:

```
Dim objGroupWise as object

Set objGroupWise = CreateObject("NovellGroupWareSession")
```

2.1.2 Log In

Next, log in to GroupWise by following the steps for

- “Logging In With Early Binding” on page 21
- “Logging In With Late Binding” on page 21

Logging In With Early Binding

For early binding, log in by invoking the Application object's (objGroupWise) Login method. From Visual Basic, use:

```
Dim objAccount as Account2

Set objAccount = objGroupWise.Login(txtUserID, txtCommandLine)
```

Your application is now logged in to GroupWise and the Application object (objGroupWise) properties are valid. Information store contents are accessible through the Application object.

Logging In With Late Binding

In late binding, you are able to use the same login method as above. However, you just need to declare objAccount as an "Object" (the type given in Visual Basic) instead of Account2. TxtUserID and txtCommandLine are optional parameters. If they are left out, GroupWise attempts to log you in by first trying your NDS login ID, then your Windows login ID, and finally by trying the information stored in the registry from your last successful login.

If you don’t need to use the GroupWise account later, you can replace the Set objGWAccount = with Call. The string txtUserID is the user ID name. The string txtCommandLine contains any valid command line parameters. You can also pass optional third and fourth parameters for the GroupWise Password and When to Prompt respectively.

Your application is now logged in to GroupWise.

2.2 Delphi

From Delphi, get a handle to the GroupWise Application object and log in to GroupWise using statements similar to Visual Basic. However, ComObj (or OleAuto for Delphi v2.*x*) must be declared in the Uses section and objGroupWise must be declared variant in the Var section.

To get the handle to the GroupWise Application object from Delphi, use:

```
objGroupWise := CreateOleObject( 'NovellGroupWareSession' );
```

To log in to GroupWise from Delphi, use:

```
objGroupWise.Login( txtUserID, txtCommandLine );
```

Important: User-defined fields that are defined in binary will function only under C++.

2.3 C ++

To get a pointer to the GroupWise Application (Session) Interface, use:

```
hResult = CoCreateInstance(CLSID_GroupWare, NULL, CLSCTX_INPROC_SERVER |
CLSCTX_INPROC_HANDLER | CLSCTX_LOCAL_SERVER, IID_IGWSession, (void **)
&pIGWSession)
```

To log in to GroupWise from C++, use:

```
pIGWSession->Login(vUserId, vCmdLine, vPassword, vWhenToPrompt, vReserved,
&pDIGWAccount)
```

To get the Root Account's Interface from the Dispatch Interface:

```
pDIGWAccount->QueryInterface(IID_IGWAccount, (LPVOID *) &pIGWAccount)
```

After importing the type library, use:

```
GroupwareTypeLibrary.Application objApplication - new
GroupwareTypeLibrary.Application();
GroupwareTypeLibrary.Account vAccount = objApplication.Login("","","","","");

MessageBox.Show(vAccount.Owner.DisplayName);
```

3 Objects

See also the following events objects that are documented as part of the GroupWise Events:

- "Event"
- "EventCollection"
- "EventConfiguration"
- "EventConfigurations"
- "EventDefinition"

This section contains information about the following objects:

Account

Allows access to objects and properties associated with one of the UserIDs logged in to or proxied for by the current Application object.

Properties

The following table lists properties along with their access and descriptions.

Property	Access	Description
AccountProperty	R/O	String. Accepts a constant from the AccountPropertiesConstants enumeration. The returned value is a string that is filled with one of the following: ◆ egwAccountPropertyDomain (String name of the GroupWise domain for the account.) ◆ egwAccountPropertyPostOffice (String name of the GroupWise Post Office for the account.) ◆ egwAccountPOAStillAlive (If the GroupWise POA is still responding to requests, Y. Otherwise, N.) ◆ egwAccountLastError (Last known GroupWise engine error code, converted to a hexadecimal string. For example, 0xD124 if access is denied.)
AccountRights	R/O	AccountRightsCollection. Access rights given to specific users. For the root account (Proxied = FALSE), contains zero or more read/write instances of AccountRights objects, each of which grants access rights to a specific user. For a proxied account (Proxied = TRUE), contains exactly one read-only instance of the AccountRights object which specifies the rights a user has to the proxied account.
AccountUID	R/O	Returns a Unique ID (UID) for the account object. For an online connection, the UID is the email address of the account. For example, jdoe@gwpo.com. For a remote or caching connection, the UID is the email address with a unique number appended to the end. For example, jdoe@gwpo.com:1456732456.
AddressBooks	R/O	AddressBooks collection. The AddressBooks collection for the root account-even if this account is a proxy account (Proxied = TRUE).
AllFolders	R/O	Folders collection. All folders in this account. Includes shared folders even if they are owned by another user (the folder's Shared property = egwSharedIncoming).
AllMessages	R/O	AllMessages collection. All messages in this account. Does not include messages that belong to other accounts, even if they appear in this account's query folders and shared folders.
Application	R/O	Application. The Application object.
Archived	R/O	Boolean. TRUE if this is an archive account.
Cabinet	R/O	Folder. The Cabinet Folder.

Property	Access	Description
Calendar	R/O	Folder. Contains the appointments, notes, and tasks stored in this account. Items stored in shared folders are not part of the Calendar folder. Items that have the OnCalendar property set to FALSE are not part of the Calendar folder.
CategoryDefinitions	R/O	CategoryDefinitions collection. The collection of Category objects defined for this account.
Checklist	R/O	Folder object. Returns an interface pointer to the Checklist folder object.
Contacts	R/O	Folder object. Returns an interface pointer to the Contacts folder object.
DefaultAccountRights	R/O	AccountRights. Default access rights given to users not listed in the AccountRights property. Only available from the root account.
DefaultAddressBook	R/W	AddressBook. Only available from the root account.
DefaultDocumentLibrary	R/O	DocumentLibrary. The default DocumentLibrary object. Can be Nothing.
DefaultPathToArchive	R/O	String. Returns the path to the archived account, if one exists.
DistinguishedName	R/O	String. Returns a string with the full eDirectory Distinguished Name for the user. For example, JDoe.MyTree.MyCompany. Because the remote mode and caching mode have no connection to the Domain or Post Office databases, this property is not valid in these modes.
DocumentLibraries	R/O	DocumentLibraries collection.
DocumentsFolder	R/O	Folder. The folder containing the most recently used documents.
FieldDefinitions	R/O	FieldDefinitions collection.
Filters	R/O	Filters collection. Represents saved filters.
FrequentContacts	R/O	AddressBook. Only available from the root account. Can be Nothing, even in the root account, if the Novell Address Book is not installed.
GWTimeZone	R/O	The GWTimeZone object for this application.
LastBackupDate	R/W	Date. The date and time this account was last backed up. You must be logged in as a Trusted Application to set this date in SOAP. -Caching or Remote modes. It does not work in Client/Server mode.
LastRetentionDate	R/W	Date. The date and time this account was last archived (retained) by a third-party archive program. You must be logged in as a Trusted Application to set this date in SOAP. -Caching or Remote modes. It does not work in Client/Server mode.
MailBox	R/O	Folder. The Mailbox folder.
MoveAcceptedAppointmentsfrom Mailbox	R/W	Boolean. True if appointments are automatically moved to the calendar folder when accepted; FALSE otherwise.

Property	Access	Description
MultiLoginAddressBookSupport	R/W	Boolean. If TRUE, accounts created by calling MultiLogin do not use MAPI address book methods to retrieve their data, which allows each MultiLogin account to see the address book data for each user. If FALSE, all accounts use the Address Book data for the first account logged into by GroupWise
ObjType	R/O	Enum (egwUser, egwResource). The type of object this account represents.
Owner	R/O	Address. The address of the owner of this account.
Parent	R/O	Application. The Application object that owns this object.
PathToArchive	R/O	String. Returns the path to the archived account, if one exists. **NOTE:** The GroupWise client does not update settings until it exits. If you are running multiple clients, there is not a general purpose "settings have changed" event mechanism in place to notify other clients to reread the settings and there is no way for the Object API to receive updated settings information that is changed by a client. The best option is to have third-party products use tokens against a running client. However, if multiple clients are running for the same user and the settings change on the other client, the settings will still be incorrect for the client that the C3PO is running against.
PathToHost	R/O	String. The path-to-host of the GroupWise server that was logged in to, or an empty string ("") if the user logged in through TCP/IP.
PeekMode	R/W	Boolean. If a third-party application has accessed this account using a Trusted Application login, this flag allows the third-party application to read email messages without setting the Read flag. Any application that created a GroupWise User account can set the PeekMode flag on that account. If PeekMode is on and you try to access the Message Body or Attachments for a specific Message, the Downloaded flag is set (and the Opened and Read flags are not automatically set). A Trusted Application can use the Account6 interface to set the flags that are automatically set during PeekMode. You can also use this interface to ensure that no Opened, Read, or Downloaded flags are set.
PeekModeFlagConstants	R/W	`Reading`: Any third-party program can get the state of the PeekModeFlags and see what status flags are set when a Message object is accessed. It does not matter if PeekMode is turned ON or not. These flags are applies to all Messages. By default, this returns a bitmask with two of the enumeration values set: ◆ egwPeekModeOpenedFlag ◆ egwPeekModeReadFlag `Writing`: Only Trusted Applications can call to change the value of the PeekMode status flags. The PeekModeFlagConstants enumeration is a bitmask, so any or all of the values can be ORed together.
Proxied	R/O	Boolean. TRUE if this is a proxy account.

Property	Access	Description
ProxyAccounts	R/O	Accounts collection. The proxy accounts currently being used.
Remote	R/O	Boolean. TRUE if this is a remote account.
RootFolder	R/O	Folder. The top-level folder that contains all other folders.
Rules	R/O	Rules. A collection of the Rules defined for the account.
Sent Items	R/O	Folder object. Returns an interface pointer to the Sent Items system folder object.
SystemAddressBook	R/O	AddressBook. Only available from the root account.
TCPIPAddress	R/O	String. The TCP/IP address of the GroupWise server that was logged in to, or an empty string ("") if the user logged in through a mapped path or a Universal Naming Convention (UNC) path.
TCPIPPort	R/O	Long. The TCP/IP port of the GroupWise server that was logged in to, or zero (0) if the user logged in through a mapped path or a UNC path.
Trash	R/O	Trash. The trash for this account.
UNCPathToDomain	R/O	String. Returns a string with the UNC Path to the Domain database for this user's account.
UsingSSL	R/O	Boolean. TRUE if this connection between the POA and client is using SSL encryption. FALSE if this connection is using regular GroupWise encryption.
VCardDisplayFlag	R/W	Boolean. TRUE if a VCard should be added to Emails sent from this account; False otherwise.
VCardPABEmailAddress	R/W	String. The email address to use inside the personal address book when generating a VCard.
VCardPABName	R/W	String. The name of the personal address bookto use when generating a VCard. Only used if VCardSource is set to VCardPABSource.
VCardSignatureFile	R/W	String. The full path to a file containing a VCard to be added to Emails sent from this account. If this string is empty and the VCardDisplayFlag is set, the GroupWise Windows Client will automaticallygenerate a VCard to send with the message.
VCardSource	R/W	VCardSourceConsistants. Points at the source that should be used to generate a VCard that is added to any sent message.
WorkFolder	R/O	Folder. The Work In Progress folder.

Methods

See the following methods in the GroupWise Events documentation:

- “getEventConfiguration”
- “getEvents”
- “removeEvents”

String ConvertEmailAddress(String OldAddress, [EmailAddressFormatConstants Format])

Converts OldAddress into a new format specified by Format, and returns it as a string. OldAddress may be either a string or an Address. If no Format is specified, it defaults to egwEmailAddress4x.

Query CreateQuery()

Creates a new Query object. See Query.

Account GetArchiveAccount([VARIANT Path])

Returns the archive account specified by Path. If no Path is specified, returns the archive account that has been set by SetArchiveTo. If SetArchiveTo is not used, this method looks up the archive account in the database. If no archive path can be found in the database, an exception is thrown. Only available from the root account.

Folder GetFolder(String FolderID)

Returns the folder in this account with the specified FolderID.

Message GetMessage(String MessageID)

Returns the message in this account with the specified MessageID.

IMPORTANT: GetMessage does not return items from the folders that are shared with you. It does return items from folders shared with others.

QuickMessages GetQuickMessagesCollection(Date StartDate, QuickMessagesCreationConstants eHowBuildList)

Returns a QuickMessages (page 147) collection object. For a list of properties available, see QuickMessage (page 145).

The eHowBuildList parameter controls which message items are included in the collection. The following values are possible for eHowBuildList:

egwNewMessages	Messages items in this account that were created or received only since the StartDate parameter are included
egwModifiedMessages	Message items in this account that have been changed only since the StartDate parameter are included
egwAllMessages	Every message item in this account is added to the QuickMessages collection, and the StartDate parameter is ignored.

The QuickMessages collection should return message items quicker than other ways (such as the AllMessages collection or querying objects with their message lists). It is very fast when returning new messages, but retrieving modified messages is somewhat slower. The slowest is to retrieve all messages from an account.

GetQuickMessagesCollection should be called only on a main account. Calling this method from a Proxy account results in an error.

Messages in a folder that have been shared do not appear in the resulting QuickMessages collection. These folders are not part of the user's database and are outside of the account.

Folder (page 101) contains a similar method. However, the Account object method uses every message in the account, while the Folder object method looks through message items that are in the specified folder.

NOTE: This method does not return hidden messages. Call the GetQuickMessagesCollectionExt method to return hidden messages.

`QuickMessages GetQuickMessagesCollectionExt (Date StartDate, QuickMessagesCreationConstants, eHowBuildList, BOOL bIncludeHidden)`

Returns a QuickMessages collection object. The bIncludeHidden parameter dictates whether Hidden messages are included with the returned collection.

The eHowBuildList parameter controls which message items are included in the collection. The following values are possible for eHowBuildList:

egwNewMessages	Messages items in this account that were created or received only since the StartDate parameter are included
egwModifiedMessages	Message items in this account that have been changed only since the StartDate parameter are included
egwAllMessages	Every message item in this account is added to the QuickMessages collection, and the StartDate parameter is ignored.

This method should return message items more quickly than other ways (such as using the AllMessages collection and querying objects with their Message Lists). Retrieving Modified Messages is slower than returning New Messages. Retrieving All Messages from an Account is the slowest.

`AddressBook GetSystemAddressBook ([VARIANT bIncludeHiddenFile])`

Returns the SystemAddressBook for this account object. If the Account object was retrieved by a Trusted Application and the MultiLoginAddressBookSupport flag is TRUE, the bIncludeHiddenFile parameter is used. Otherwise, this method works as if the normal SystemAddressBook property was called.

Asking for the SystemAddressBook and then retrieving AddressBookEntries excludes entries that are marked in ConsoleOne with Visibility - None or with Visibility set for their local Post Office only.

Trusted Applications, however, can view Restricted AddressBookEntries if they call this method and set the boolean property inside the bIncludeHidden parameter to TRUE.

The bIncludeHidden parameter is option. If the parameter is not filled with a valid boolean value, the property behaves as if the parameter is set to FALSE and all hidden entries are excluded from the list.

`MergeArchive(Account DestArchiveAcct)`

Merges all messages from the current archive account into the archive account specified by DestArchiveAcct.

`Account Proxy(VARIANT User)`

Returns the proxy account specified by User. User may be either a string or an Address object. If User is a string, it represents the UserID of the desired proxy account. If User is an Address object, it represents the address of the desired proxy account.

`Refresh()`

Forces this Account object, and all objects and collections owned by this account, to reread property values from the message database.

SetArchiveTo([VARIANT Path])

Changes the archive path to the path specified by Path. If no Path is specified, this method will change the archive path to the archive path specified in your GroupWise preferences. Only available from the root account.

SetPassword(String OldPassword, String NewPassword)

Changes the current user's GroupWise password from OldPassword to NewPassword. If the OldPassword is not the current GroupWise password, an error will be returned. If the GroupWise account has no password, the OldPassword should be an empty string. Otherwise, an error will be returned. Another error will be returned if the user's Post Office has been set to LDAP authentication.

NOTE: GroupWise cannot change the user's NetWare, Windows, LDAP, or other passwords. GroupWise changes only the password associated with the current GroupWise database. For example, if you are logged in using a TCP/IP address (which is also called online mode), SetPassword changes the password used to login to the TCP/IP address. If you are logged in to a remote or client cache database, SetPassword will change the password used to login to this remote or client cache box. SetPassword will not synchronize an Online (master) password to a remote or client/cache database (and vice versa).

SynchronizeToRemote(String path, String masterPassword, Boolean isCache, SynchronizeConstants flags);

Performs the equivalent of the regular client's "Hit The Road" functionality. path is the desired name for the remote mailbox or the client cache mailbox. masterPassword is the password of the user's main mailbox. isCache indicates if the call is setting up regular remote or client cache.

The flags are used to specify what parts to download (see Section 4.46, “SynchronizeConstants,” on page 190).

SynchronizeWithMaster(SynchronizeConstants flags);

Synchronizes the user's master mailbox in remote mode. The user needs to have set up the connection to the master post office from the regular client or by using `SynchronizeToRemote`.

The flags specify what parts to synchronize (see Section 4.46, “SynchronizeConstants,” on page 190).

AcceptToFolder(Folder gwDestFolder, [string Comment],[Boolean bAllInstances]);

Accept this appointment and place it in the given destination folder. If the destination folder is not a personal folder, the mailbox folder, the calendar folder, the cabinet folder, the Work In Progress folder, or a sub-calendar folder, an error will be returned. Further, the destination folder can not be a shared folder. For example a folder shared with the user. If the optional comment is provided, it will be returned to the appointment sender. If this appointment is part of an Auto Date series and the optional bAllInstances flag is set to TRUE, all instances of this appointment will be accepted.

AccountRights

Provides access rights to an account.

Properties

The following table lists properties along with their access and descriptions.

Property	Access	Description
Address	R/O	Address. The address of the user to whom access rights are being given.
Application	R/O	Application. The Application object.
Archive	R/W	Boolean. TRUE if the user can archive messages.
BitMask	R/W	Long. A bit mask that contains one bit for each Boolean property. See AccountRightsConstants for the bit definitions. This property is synchronized with the Boolean properties.
ModifyPrefsRulesGroups	R/W	Boolean. TRUE if the user can modify preferences, rules, and groups.
Parent	R/O	Object. The AccountRightsCollection or the Account object that owns this object.
ReadAppointments	R/W	Boolean. TRUE if the user can read appointments.
ReadMailPhone	R/W	Boolean. TRUE if the user can read mail and phone messages.
ReadNotes	R/W	Boolean. TRUE if the user can read notes.
ReadPrivate	R/W	Boolean. TRUE if the user can read private messages.
ReadTasks	R/W	Boolean. TRUE if the user can read tasks.
ReceiveAlarms	R/W	Boolean. TRUE if the user can receive alarms.
ReceiveNotifications	R/W	Boolean. TRUE if the user can receive notifications.
WriteAppointments	R/W	Boolean. TRUE if the user can write appointments.
WriteMailPhone	R/W	Boolean. TRUE if the user can write mail and phone messages.
WriteNotes	R/W	Boolean. TRUE if the user can write notes.
WriteTasks	R/W	Boolean. TRUE if the user can write tasks.

Methods

`Delete()`

Deletes the user's access rights to the account.

Remarks

The properties of the AccountRights object can only be written by the owner of the account.

An AccountRights object is refreshed when its parent object is refreshed. When an AccountRights object is refreshed, it recursively refreshes its Address object.

AccountRightsCollection

A collection of AccountRights objects.

Properties

The following table lists properties along with their access and descriptions.

Property	Access	Description
Application	R/O	Application. The Application object.
Count	R/O	Long. The number of objects in this collection.
_NewEnum	R/O	Enumeration object. Implements IEnumVARIANT. For Windows only.
Parent	R/O	Account. The Account object that owns this collection.

Methods

`Add(VARIANT Address, Long Rights)`

If Address is an Address object, this method creates a new AccountRights object for that address. If Address is an Addresses collection, this method creates a new AccountRights object for each address in the collection. Rights specifies the rights to be given. The Rights value is derived from the AccountRightsConstants combined with the bit-wise inclusive OR operator. This method does not return the newly created object.

`AccountRights Item(Long Index)`

DEFAULT. Returns the AccountRights object located at the given Index in the collection. Valid indexes are 1 through Count, inclusive. Throws an exception if the Index is outside of this range.

Remarks

An AccountRightsCollection is refreshed when its parent object is refreshed. When an AccountRightsCollection is refreshed, it recursively refreshes its contained AccountRights objects.

Accounts

A collection of Account objects.

Properties

The following table lists properties along with their access and descriptions.

Property	Access	Description
Application	R/O	Application. The Application object.
Count	R/O	Long. The number of objects in this collection.
_NewEnum	R/O	Enumeration object. Implements IEnumVARIANT. For Windows only.
Parent	R/O	Application. The Application object that owns this collection. This property is included for completeness and is identical to the Application property.

Methods

`Account Item(Long Index)`

DEFAULT. Returns the Account object located at the given Index in the collection. Valid indexes are 1 through Count, inclusive. Throws an exception if the Index is outside of this range.

Remarks

The Accounts collection has no direct Add method. Instead, Account objects are added by the Application object's Proxy method, or by the Application object's MultiLogin method.

There is no need to refresh an Accounts collection, it is always up-to-date.

Address

Describes the address of a user, group, or resource.

Properties

The following table lists properties along with their access and descriptions.

Property	Access	Description
Application	R/O	Application. The Application object.
DisplayName	R/O	String. The descriptive name to be displayed to users.
EmailAddress	R/O	DEFAULT. String. The email address the system uses to deliver mail. The format is determined by EmailType.
EmailType	R/O	String. The email address type. "NGW" indicates an internal GroupWise address. Anything else is an external address. External addresses are submitted to the operating system's default email transport. (In Windows, for example, this property would be MAPI.)
ObjType	R/O	Enum (AddressTypeConstants). The type of entity this Address represents.
Parent	R/O	Object. The object that owns this Address. There are many possible owner objects, although any specified Address object will have a single owner.

Remarks

An Address object is refreshed when its parent object is refreshed.

Each Address object originates from a resolved address which is either a GroupWise address from an address book or an address external to GroupWise (such as an Internet address). A user who has been deleted from GroupWise may have an unresolvable address, in which case an Address object will represent the sender even if the sender is subsequently deleted from GroupWise.

AddressBook

Provides access to address book data.

Properties

The following table lists properties along with their access and descriptions.

Property	Access	Description
AddressBookEntries	R/O	AddressBookEntries collection. The collection of entries in this address book.
AddressBookRights	R/O	AddressBookRights collection. Describes the rights users have to this address book.
Application	R/O	Application. The Application object.
Default	R/O	Boolean. TRUE if this is the default address book.
FieldDefinitions	R/O	FieldDefinitions collection. Defines the user-defined and predefined fields that can appear in this address book.
Name	R/O	String. The name of this address book.
ObjType	R/O	Enum (AddressBookTypeConstants) The type of this address book.
Owner	R/O	Address. Returns the address of the owner of the address book. **--GroupWise 8 SP1 and later.**
Parent	R/O	AddressBooks collection. The AddressBooks collection that owns this object.
ProviderID	R/O	String. A string identifying the address book's provider. Typically a Globally Unique Identifier (GUID) converted to a hex string. An empty string ("") means the provider is unidentified.
Shared	R/O	Enum (SharedFolderConstants). Returns whether the address book is shared.
SortOrder	R/W	Enum (SortConstants). Specifies whether the AddressBookEntries collection returns entries in ascending or descending order. This property is not persistent. It can be set, but it will revert back to the default value set by the address book provider when the AddressBook object is refreshed or freed from memory.
Subtype	R/O	Enum (AddressBookSubtypeConstants). The subtype of this address book.

Methods

`Delete()`

Deletes this address book from the parent AddressBooks collection.

`Refresh()`

Forces this AddressBook object, and all objects and collections owned by this address book, to reread property values from the message database.

`Rename(String Name)`

Renames the address book to the new name.

Remarks

Creating a new personal address book will automatically add a new folder under the contacts folder by the same name. Similarly, deleting a personal address book will remove the corresponding user contacts folder. If the deleted personal address book was currently assigned to the main contact folder, the GroupWise system will chose another personal address book to replace it in the contacts folder.

AddressBookEntries

A collection of AddressBookEntry objects.

Properties

The following table lists properties along with their access and descriptions.

Property	Access	Description
Application	R/O	Application. The Application object.
Count	R/O	Long. The number of objects in this collection. The system address book does not keep an actual count of the number of users in the book. The count that is returned is an approximate count that is based on an index count. Since the index is not updated with every delete (for performance reasons), items that have been deleted are still included in the count until the index is rebuilt. Hidden address book entries can also be included in the count. If you want an exact count, use the AddressBook (page 37) object to iterate through every entry in the book and count each iteration. **WARNING:** To keep your system from crashing, test for a NULL value in the returned entry before using the object.
_NewEnum	R/O	Enumeration object. Implements IEnumVARIANT. For Windows only.
Parent	R/O	AddressBook. The AddressBook object that owns this collection.

Methods

`AddressBookEntry Add(VARIANT Address, [AddressTypeConstants ObjType])`

Adds a new AddressBookEntry to the collection based on an existing address. The Address parameter must be an object of type Address or AddressBookEntry. The new entry's DisplayName, EmailAddress, and EmailType are copied from Address. If ObjType is omitted, egwUser is assumed. See Remarks below.

`AddressBookEntry Add(String DisplayName,[String EmailAddress],[String EmailType], [AddressTypeConstants ObjType])`

Adds a new AddressBookEntry to the collection based on DisplayName. The DisplayName parameter must be of type String. If EmailAddress is omitted, an empty string ("") is assumed. If EmailType is omitted, "NGW" is assumed, except when EmailAddress is an empty string, in which case an empty string is assumed for EmailType. If ObjType is omitted, egwUser is assumed. See Remarks below.

`Addresses Find(String Condition)`

Returns a new Addresses collection containing the addresses from this collection that match the filter expression given by the Condition parameter (see Chapter 5, “Filter Expressions,” on page 195). This Find method differs from the Find methods for messages in that it accepts a restricted syntax and does not accept a saved Filter. Each address book may support a different set of filtering operations. Filtering on an address book that supports a limited set of operations will throw an exception when handed a filter expression beyond the address book's capabilities. This method does not support BEGINSWITH, but it does support MATCHES and CONTAINS.

You can search only on First Name and Last Name in the System Address Book. If you try searching on Name, this method fails.

AddressBookEntry Item(VARIANT Index)

DEFAULT. Returns the AddressBookEntry object specified by Index. Index may be a Long, a string representing a Long, or an Address object. If Index is a Long, or a string representing a Long, returns the AddressBookEntry object located at the given Index in the collection. Valid indexes are 1 through Count, inclusive. Throws an exception if the Index is outside of this range. If Index is an Address object, it should contain the DisplayName, EmailAddress, EmailType, and ObjType of the desired AddressBookEntry object.

AddressBookEntry ItemByGuid(BSTR bstrGuid)

Accepts a Bstring value that holds the GUID for an Address Book Entry. A GUID is available only if the MultiLoginAddressBookSupport flag is turned ON. To get a GUID, access the AddresBookEntry objects Fields collection. Iterate trhough the Fields collection looking for a field named PABGuid. which is for Personal Address Book Entries, or GUID, which is for System Address Book Entries. This method returns the AddressBookEntry that matches the GUID. If no entry matches the GUID, a warning of S_FALSE is returned and the AddressBookEntry object pointer is NULL.

Remarks

IMPORTANT: The Add method can be called only when adding an address book entry to a personal address book. It cannot be used with the System or LDAP address books.

The Add method template is as follows:

Add(VARIANT P1, [VARIANT P2], [VARIANT P3], [VARIANT P4])

The Add method checks the P1 type at runtime.

- If P1 is an Address object or an AddressBookEntry object, the first form of Add is assumed; P2, if present, must be an ObjType Enum of type AddressTypeConstants, and P3 and P4 must not be present.
- If P1 is a String, the second form of Add is assumed; P2 and P3, if present, must be Strings and P4, if present, must be an ObjType Enum of type AddressTypeConstants.

An AddressBookEntries collection is refreshed when its parent object is refreshed. When an AddressBookEntries collection is refreshed, it recursively refreshes its contained AddressBookEntry objects.

AddressBookEntry

Describes an entry in an address book. A subtype of Address. Allows you to modify the properties of an address book entry.

Properties

The following table lists properties along with their access and descriptions.

Property	Access	Description
Application	R/O	Application. The Application object.
Categories	R/O	Categories Collection. The collection of Category objects on this Mail, Appointment, Document Reference, Note, Phone Message, Task, or AddressBookEntry.
DisplayName	R/W	String. The descriptive name to be displayed to users. This is a required field. If there are entries in the Address Book without a display name, running "GWCHECK Contents/Fix" against the user database will generate a diplay name for each entry. It will first attempt to generate the display name by using the First and Last name. If these fields are blank, it will use the email address.
DefaultMailingAddress	R/W	AddressBookDefaultMailingAddressConstants. Which postal mailing address to use as the default. Foe example, use the bussiness address for this contact as the default when sending mail.
EmailAddress	R/W	DEFAULT. String. The email address the system uses to deliver mail. The format is determined by EmailType.
EMailAddresses	R/O	EMailAddresses Collection. The collection of Email Addresses on this Address Book Entry. This property should be used only for personal address books. Using it for the System Address Book results in an error.
EmailType	R/W	String. The type of email address. "NGW" indicates an internal GroupWise address. Anything else is an external address. External addresses are submitted to the operating system's default email transport. (In Windows, for example, this property would be MAPI.)
Fields	R/O	Fields collection. The fields for this entry. GroupWise supports fields on simple address book entries. Address book entries that are groups do not support the Fields collection.
HasThumbnailImageToJPGFile	R/O	Boolean. True if this AddressBook has an associated JPG Thumbnail image; FALSE otherwise.
IMAddresses	R/O	IMAddresses Collection. The collection of Instant Messaging Addresses on this Address Book Entry.
MapiEntryID	R/O	Safe array inside a variant structure. This property is available only using C. The internal C++ code takes a pointer to a variant. Once the variant is received, the calling code must check the variant type and then access the appropriate piece of the variant structure to retrieve the data. In C++, this property calls SafeArrayAccessData and SafeArrayUnaccessData.
MasterRevisionNumber	R/O	Long. The revision number of this entry when it was downloaded to remote.

Property	Access	Description
Members	R/O	GroupMembers collection. If this entry is for a group (ObjType = egwGroup), this object returns the members of the group. Otherwise, it returns an empty collection.
ObjType	R/W	Enum (AddressTypeConstants). Type of entity this entry represents.
Parent	R/O	AddressBookEntries collection. The AddressBookEntries collection that owns this object.
RevisionNumber	R/O	Long. The number of times this entry has been changed.

Methods

`Delete()`

Deletes this address book entry from the parent AddressBookEntries collection, and from the address book. Deleting an AddressBookEntry from certain address book providers is not allowed and may throw an exception. The GroupWise system address book does not allow deletions.

`Refresh()`

Forces this AddressBookEntry object, and associated Fields, to reread property values from the message database. An AddressBookEntry object is refreshed when its refresh method is called or its parent object is refreshed.

`RemoveThumbnailImage()`

If this AddressBookEntry has an associated thumbnail, removes the image from the AddressBookEntry.

SetImageAsJPGThumbnail(String strFileName)

Takes the image file given in strFileName anf converts it to a JPG Thumbnail image, then stores the thumbnail in the AddressBookEntry.

Image files may be a JPG, GIF, BMP, PNG, TIFF, WMF, or ICON type.

`SaveThumbnailImageToJPGFile(String strFileName)`

If this AddressBookENtry has an associated thumbnail, it saves the image to the given file name.

NOTE: All thumbnail images are stored in JPG format. It is strongly suggested that the given file name end with a ".jpg" extension.

Fields

There are three birthday fields:

- BIRTHDAY_DAY
- BIRTHDAY_MONTH
- BIRTHDAY_YEAR

These statements are numeric filter expressions. For example, BIRTHDAY_MONTH=4 finds all Address Book entries with a birthday in the month of April. To allow for broader searching capabilities, these fields were purposely split into the three sections.

AddressBookRights

Describes the access rights a user has to an address book.

Properties

The following table lists properties along with their access and descriptions.

Property	Access	Description
Access	R/W	Enum (AddressBookRightsConstants). The access rights a user has to an AddressBook.
Address	R/O	Address. The address of the user.
Application	R/O	Application. The Application object.
Parent	R/O	AddressBookRightsCollection. The AddressBookRightsCollection that owns this object.

Methods

`Delete()`

Delete the user's access rights to the address book.

AddressBookRightsCollection

A collection of AddressBookRights objects.

Properties

The following table lists properties along with their access and descriptions.

Property	Access	Description
Application	R/O	Application. The Application object.
Count	R/O	Long. The number of objects in this collection.
_NewEnum	R/O	Enumeration object. Implements IEnumVARIANT. For Windows only.
Parent	R/O	AddressBook. The AddressBook object that owns this collection.

Methods

`AddressBookRights Add(VARIANT Address, AddressBookRightsConstants AccessRights)`

If Address is an Address object or an AddressBookEntry object, this method creates a new AddressBookRights object for that address. If Address is an Addresses collection, this method creates a new AddressBookRights object for each address in the collection. AccessRights specifies the rights given. See AddressBookRightsConstants.

`Commit(String Subject, String BodyText)`

"Shares" the address book with the users in the collection, and sends a message to them, with the given Subject and BodyText, notifying them that the address book has been shared with them.

`AddressBookRights Item(VARIANT Index)`

DEFAULT. Returns the AddressBookRights object specified by Index. Index may be a Long, an Address object, or an AddressBookEntry object. If Index is a Long, returns the AddressBookRights object located at the given Index in the collection. Valid indexes are 1 through Count, inclusive. Throws an exception if the Index is outside of this range. If Index is an Address object or an AddressBookEntry object, it should contain the DisplayName, EmailAddress, EmailType, and ObjType of the desired AddressBookRights object.

Remarks

An AddressBookRightsCollection is refreshed when its parent object is refreshed.

AddressBooks

A collection of AddressBook objects.

Properties

The following table lists properties along with their access and descriptions.

Property	Access	Description
Application	R/O	Application. The Application object.
Count	R/O	Long. The number of objects in this collection.
_NewEnum	R/O	Enumeration object. Implements IEnumVARIANT. For Windows only.
Parent	R/O	Account. The Account object that owns this collection.

Methods

`AddressBook Add(String Name)`

Creates a new personal address book with the given Name.

`AddressBook AddEx(SharedNotification SharedAddressBookNotify, [VARIANT Name])`

Accepts a new shared personal address book as specified in the SharedAddressBookNotify parameter, which is a subtype of a Message object. You can also supply an optional Name for the shared address book. If Name is omitted, the Name property of SharedAddressBookNotify will be used.

`AddressBook Item(VARIANT Index)`

DEFAULT. Returns the AddressBook object specified by Index. Index may be either a Long or a string. If Index is a Long, returns the AddressBook object located at the given Index in the collection. Valid indexes are 1 through Count, inclusive. Throws an exception if the Index is outside of this range. If Index is a string, it represents the name of the desired AddressBook object.

Remarks

An AddressBooks collection is refreshed when its parent object is refreshed. When an AddressBooks collection is refreshed it updates which AddressBook objects are in the collection, but does not recursively refresh the AddressBook objects themselves.

Addresses

A collection of Address objects.

Properties

The following table lists properties along with their access and descriptions.

Property	Access	Description
Application	R/O	Application. The Application object.
Count	R/O	Long. The number of objects in this collection.
_NewEnu m	R/O	Enumeration object. Implements IEnumVARIANT. For Windows only.
Parent	R/O	Object. The Application object or the AddressBookEntries collection that owns this collection.

Methods

`Add(VARIANT Address)`

Adds an existing address or list of addresses to this collection. Address may be an Address object, an AddressBookEntry object, or an Addresses collection. If Address is an Addresses collection, each Address in that collection is added.

`Address Item(VARIANT Index)`

DEFAULT. Returns the Address object specified by Index. Index may be a Long, a string representing a Long, or an Address object. If Index is a Long or a string representing a Long, returns the Address object located at the given Index in the collection. Valid indexes are 1 through Count, inclusive. Throws an exception if the Index is outside of this range. If Index is an Address object, it should contain the DisplayName, EmailAddress, EmailType, and ObjType of the desired Address object. Depending on the amount of supplied information, the specified address can be ambiguous, in which case the first item that is considered a match is returned.

`Remove(VARIANT Index)`

Removes the address, specified by Index, from the collection. Index may be a Long, a string representing a Long, or an Address object. If Index is a Long, or a string representing a Long, removes the Address object located at the given Index in the collection. Valid indexes are 1 through Count, inclusive. Throws an exception if the Index is outside of this range. If Index is an Address object, it should contain the DisplayName, EmailAddress, EmailType, and ObjType of the desired Address object.

Remarks

An Addresses collection is refreshed when its parent object is refreshed. When an Addresses collection is refreshed, it recursively refreshes its contained Address objects.

AllMessages

Contains all messages in an account.

Properties

The following table lists properties along with their access and descriptions.

Property	Access	Description
Application	R/O	Application. The Application object.
_NewEnum	R/O	Enumeration object. Implements IEnumVARIANT. For Windows only.
Parent	R/O	Account. The Account object that owns this collection.

Methods

`AllMessagesIterator CreateAllMessagesIterator()`

Creates an AllMessagesIterator object for iterating over the messages in this collection.

`MessageList Find(VARIANT Condition)`

Returns a list of messages matching the Condition specified. Condition may be either a string or a Filter object. If Condition is a string, it represents a filter expression. See Chapter 5, “Filter Expressions,” on page 195. If Condition is a Filter object, it represents a saved Filter. To save the overhead of instantiating a messages collection, callers of the API should use the Find method in the AllMessages collection if they do not already have a messages collection or when searching across folders. Once a message collection has been obtained, there is no loss of efficiency to call its Find method.

Remarks

Because AllMessages is a large collection, it does not support a Count property or an Item method. You can access its elements using an iterator object or using the GetMessage method of the parent Account object to access a message by MessageID.

An AllMessages collection is refreshed when its parent object is refreshed. When an AllMessages collection is refreshed, it updates the Message objects in the collection, but it does not recursively refresh the Message objects themselves

AllMessagesIterator

Iterates over the messages in an AllMessages collection, successively returning each message. Multiple AllMessagesIterator objects can be simultaneously active over the same collection. Each iterator maintains its own internal position and does not affect the state of the collection it is iterating.

Properties

The following table lists properties along with their access and descriptions.

Property	Access	Description
Application	R/O	Application. The Application object.
Parent	R/O	AllMessages collection. The collection over which this AllMessageIterator iterates.

Methods

`AllMessagesIterator Clone()`

Creates another, independent AllMessagesIterator object. The clone starts at the original object's current position in the collection. Use Clone() to save the current position before a Skip or Reset.

`Message Next()`

Returns the next message in the collection or nothing after the last message.

`Reset()`

Resets an iterator so that Next returns the first message in the collection.

`Skip(Long NumMessages)`

Skips the next NumMessages messages. Skip does nothing if there are not NumMessages messages left to skip, or if NumMessages < 0.

Remarks

Both AllMessagesIterator and IEnumVARIANT (from the AllMessages collection _NewEnum method) allow iterating an AllMessages collection. However, AllMessagesIterator is an object available in all object-oriented development environments, while IEnumVARIANT is a COM interface inaccessible in environments such as Delphi. The usual alternative to IEnumVARIANT is:

```
for i:= 1 to Count
```

This does not work with large collections, such as AllMessages, which do not support Count or Item.

The AllMessagesIterator object does not support a Refresh method.

Application

The root Application object. All other GroupWise objects are contained by this object.

Properties

The following table lists properties along with their access and descriptions.

Property	Access	Description
Application	R/O	Application. The Application object; provided for consistency.
BuildNumber	R/O	A number that identifies different versions of Field Test File (FTF) patches and indicates when a fix was included. For example, if you receive two FTF updates for version 6.5.4, you can tell the difference between the two files by the BuildNumber.
CommandLine	R/O	String. The command line that was used in the Login method to log in to the root account. Returns an empty string ("") if not logged in.
FullName	R/O	String. Fully-qualified path\file name of Windows executable.
LoginError	R/O	String. The LoginError property will return the engine error that caused the problem.
MainAccounts	R/O	Accounts collection. The accounts logged into through the Login or MultiLogin methods.
MajorVersion	R/O	Long. The major number (the number before the period) in the application's version number. Example: In the application version number 2.1, 2 is the MajorVersion.
MinorVersion	R/O	Long. The minor number (the number after the period) in the application's version number. Example: In the application version number 2.1, 1 is the MinorVersion.
MultiLoginAddressBookSupport	R/W	Boolean. If TRUE, accounts created by calling MultiLogin do not use MAPI address book methods to retrieve their data, which allows each MultiLogin account to see the address book data for each user. If FALSE, all accounts use the Address Book data for the first account logged into by GroupWise.
Name	R/O	DEFAULT. String. The name of this application.
Parent	R/O	Application. A reference to this object; provided for consistency.
ProxyAccounts	R/O	Accounts collection. The proxy accounts currently being used.
ProxyHistory	R/O	Addresses collection. The list of all addresses to which the root user has proxied.
RemoteAccount	R/O	Account. The remote account that corresponds to the root account. Returns nothing if (1) the root account is not a master account whose remote account is currently accessible, or if (2) the root account is not logged in.
RootAccount	R/O	Account. The root account. Returns nothing if the account is not logged in.
VersionRevision	R/O	Represents the service path revision number and is an extension of the MajorVersion and MinorVersion properties. For example, if your GroupWise version number is 6.5.4, 6 is the MajorVersion, 5 is the MinorVersion, and 4 is the VersionRevision.

Methods

`Account Login([String UserID], [String CommandLine], [String Password], [LoginConstants WhenToPrompt], [VARIANT Reserved])`

Logs in to the root account. Returns the root Account object. When an exception is thrown, returns the object pointer set to NULL. Adds the root account to the MainAccounts collection. Searches the operating system and NetWare for login information, such as user ID and password, not provided in parameters.

- When you omit WhenToPrompt or when WhenToPrompt = egwPromptIfNeeded and the Login search returns no information, Login displays a dialog box for the user. These are the only circumstances in which the Object API provides a user interface.
- When WhenToPrompt = egwNeverPrompt, Login throws an exception instead of displaying a user dialog box. Always omit the Reserved parameter.

To log in as a different user, you must either release all currently held objects, or use the MultiLogin method.

`Account MultiLogin(String UserID, String CommandLine, String Password, LoginConstants WhenToPrompt, VARIANT Reserved)`

Lets you to login to multiple accounts with the same application. Logs in to the account specified by UserID. Returns the Account object. When an exception is thrown, returns the object pointer set to NULL. Adds the account to the MainAccounts collection. It does not affect the root login, so Notify, the GroupWise client, and other such applications are not automatically logged in, and the RootAccount object is empty. The parameters for MultiLogin are the same as the parameters for Login, except that the UserID and Password parameters are both required. The one exception to this is when a Trusted Application is used. The MultiLogin method requires only a UserID with a Trusted Application. Use the Account object's Refresh method to see changes made to a MultiLogin account. Changes made to one MultiLogin account are not automatically updated in other MultiLogin accounts.

A Trusted Application can be installed by a GroupWise Administrator. Trusted Applications can use the MultiLogin method to attach to any user's online mail box. (Trusted Applications cannot automatically login to a Remote or Client/Cache mailbox.) This Trusted Application is given an Application Name and an Application Key by the GroupWise Administration system.

`Account Proxy(VARIANT User)`

Proxies as another user. Returns the proxy Account object. When an exception is thrown, returns the object pointer set to NULL. Adds the new proxy account to the ProxyAccounts collection. User may be a UserID string, or a user’s Address object.

SetTrustedApplicationCredentials(String TrustedAppName, String TrustedAppKey)

Sets the Trusted Application name and Trusted Application Key into the Object API.

A Trusted Application can be installed by a GroupWise Administrator. As part of this installation, a Trusted Application Name is passed to the GroupWise system, which returns a Trusted Application Key as a string. The Trusted Application must store this Trusted Application Name and Key pair.

NOTE: The Key is a 65-byte string (64 bytes of data and a NULL byte).

When an application wants to become a Trusted Application by using the Object API, that application provides the Trusted Application Name and Key to the Object API by calling SetTrustedApplicationCredentials.

Once an application supplies the Trusted Application, it can call the MultiLogin method to attach to any users' on-line mail box. (Trusted Applications cannot automatically log in to a Remote or Client/Cache mailbox.) The vUserID and vCommandLine parameters are filled in with the appropriate commands. If a Password is provided (using either the vPassword parameter or the command line), it must be the correct password to access this box. If a password is not provided, the Object API attempts to log in to this users box using the Trusted Application Name and Key. The GroupWise engine verifies the Trusted Application credentials for this machine, including the Trusted Application Name and Key. If the Trusted Application credentials are certified, the application may access the user's database as if they were the user themselves.

For example, if my Trusted Application was Named EncryptMessages and was given a key of 1234567890, the application calls the SetTrustedApplicationCredentials method (using C++ syntax) as in the following:

```
gw.Application->SetTrustedApplicationCredentials ("EncryptMessages",
"1234567890");
```

The Application then logs in to user JDoe by calling MultiLogin as in the following:

```
gw.Application->MultiLogin("JDoe", "/ipa-199.99.99.99 /ipp-1677", NULL,
egwPromptNever, NULL, &dispAccount);
```

If the Application Name and Application Key are found in the GroupWise Post Office database, the Trusted Application is allowed to connect to JDoe's mailbox with full access to create messages, read items, etc.

Remarks

The Application object does not need to be refreshed because it is always up to date.

Appointment

Provides appointment information and actions. A subtype of Message.

Properties

The following table lists properties along with their access and descriptions.

Property	Access	Description
Accepted	R/O	Boolean. TRUE if this appointment has been accepted.
AlarmProgram	R/W	String. The name of the program to execute when the AlarmTime has arrived.
AlarmReminderMinutes	R/W	Long. The number of minutes before the alarm rings. This property can be set even if the AlarmSet property is FALSE.
AlarmSet	R/W	Boolean. TRUE if an alarm was set for this appointment.
AlarmTime	R/W	Date. The time at which the AlarmProgram is to execute.
AllDay	R/W	Boolean. An appointment that lasts all day, so no start time is necessary. AllDay can be read on any type of appointment. It can be set on draft or personal appointments only. An AllDay appointment starts at midnight of the first day and ends at midnight of the next day. For example, the properties show up in the Object API StartDate as 23-June-2012 at 12:00:00 a.m. and in the EndDate as 24-June-2012 at 12:00:00 a.m.
Archived	R/W	Boolean. TRUE if this message has been archived.
Autodate	R/O	Boolean. TRUE if this appointment has an auto-date.
AutoDateKey	R/O	String. An ID that can be used to match all occurrences of this appointment. If this appointment is not part of a recurring series, this property returns an error.
AutodateMessages	R/O	MessageList. The list of auto-dated items. This list includes the current appointment.
BusySearchResult	R/O	BusySearchResult. The busy search results for a draft appointment. Returns nothing if StartBusySearch has never been called, or if this is a draft appointment (BoxType = egwDraft).
BusyType	R/W	Enum (TimeBlockConstants). The type of time block this appointment represents.
Categories	R/O	Categories Collection. The collection of Category objects on this Mail, Appointment, Document Reference, Note, Phone Message, Task, or AddressBookEntry.
ChecklistCompleted	R/W	Boolean. TRUE if the item is part of the Checklist folder Messages collection or a Task item AND the item has been marked Completed. FALSE otherwise.
ChecklistCompletedDate	R/W	Date. The date this item was marked completed.
ChecklistDueDate	R/W	Date. The date this item should be Completed. Returns an error if the Message item is not a Task or is not part of the Checklist folder Messages collection.

Property	Access	Description
ChecklistPercentComplete	R/W	Integer. Between 0 and 100 percent, how complete the checklist item is. If this value is set to a number outside the 0 to 100 range, an error is returned. If the value is set to 100, the item will also be marked complete.
ChecklistShowOnCalendarDate	R/W	Date. The date this item should appear on the calendar task list.
CommonMessageID	R/O	String. The ID of the message, regardless of its location. For example, an email will have the same CommonMessageID for both Sent and Received.
DelayedDeliveryDate	R/W	Date. When the Message.Send() method is called, the message is not delivered until the specified Date. This property can be used only on a Draft Message.
Delegated	R/O	Boolean. TRUE if this appointment has been delegated.
DeliveredDate	R/O	Date. The date and time this message item was delivered to the account.
DraftAutoDates	R/O	DraftAutoDates collection. The collection contains additional Dates which the Appointment, Note, or Task is created on.
Duration	R/W	Double. The duration, in days, of the appointment. For example, 0.5 = 12 hours, 1/24 = 1 hour, etc. Gives accuracy to microseconds.
EndDate	R/W	Date. The date and time when the appointment will end.
Forwarded	R/O	Boolean. TRUE if this item has been forwarded to another user. Otherwise, FALSE.
MessageProperty	R/O	String. Based on the MessagePropertiesConstant passed in, a different string is returned.
NotifywhenAccepted	R/W	Enum (NotifyMessageConstants) for more information. Specifies the type of notification to send when this appointment has been accepted.
NotifyWhenDeclined	R/W	Enum (NotifyMessageConstants) for more information. Specifies the type of notification to send when this appointment has been declined.
OnCalendar	R/W	Boolean. TRUE if this appointment should appear on a calendar display.
Personal	R/O	Boolean. TRUE if this appointment is a personal appointment.
PersonalSubject	R/W	String. A text string to be shown by the GroupWise Windows client as the Subject text. This text appears only for the current user. The normal subject or their own Personal Subject text appears for other users.
Place	R/W	String. The location of this appointment.
Replied	R/O	Boolean. TRUE if the item has been replied to. Otherwise, FALSE.
Sharer	R/O	Address. The user who added this message to a shared folder. If this message was not added to a shared folder, this will be the same as the Sender property.

Property	Access	Description
ShowOnChecklist	R/W	Boolean. TRUE if the item is part of the Checklist folder Messages collection. FALSE otherwise.
Size	R/O	Returns the size of the attachments and message body. If there are no attachments, only the size of the message body is returned.
StartDate	R/W	Date. The date and time the appointment will start. For more information about the format of this property, see Remarks.

Methods

`Accept([String Comment], [Boolean AllInstances])`

Accepts this appointment.

- Comment is the text you would like to send in reply as you accept the appointment.
- AllInstances is used for auto-date appointments. If AllInstances is set to TRUE, this method will accept all instances of the auto-dated appointment. Passing a value for AllInstances for an appointment that is not auto-dated is legal but ignored. If AllInstances is omitted, FALSE is assumed.

`Appointment Delegate([Boolean AllInstances])`

AllInstances is used for auto-date appointments. If AllInstances is set to TRUE, this method will delegate all instances of the auto-dated appointment. Passing a value for AllInstances for an appointment that is not auto-dated is legal but ignored.

`BusySearchResult StartBusySearch([Date StartDate], [Long Range])`

Starts a busy search on appointment recipients. StartDate is the starting date of the search; the time is ignored. When omitted, the appointment start date is used. Range is the number of days to search (must be greater than or equal to 7). Omitted parameters use the user's Busy Search preference settings. Any previous BusySearchResult object for this appointment is terminated and a new one created and returned.

`Decline([String Comment], [Boolean AllInstances])`

Declines this appointment. Comment is the text you would like to send in reply as you decline the appointment. AllInstances is used for auto-date appointments. If AllInstances is set to TRUE, this method will decline all instances of the auto-dated appointment. Passing a value for AllInstances for an appointment that is not auto-dated is legal but ignored. If AllInstances is omitted, FALSE is assumed.

`Double GetDateAdjustment(String Timezone)`

Returns a time zone adjustment value for the appointment. This adjustment value varies over a range of (-0.5 to 0.5 days). Most appointments return zero for this value. The return value from this method can be added directly to the StartDate and EndDate properties to account for time-zone-neutral appointments.

`LocalDelete()`

Deletes this message only from the currently connected post office database. No synchronization will take place between the master database and a remote database.

`RemoveChecklistDueDate()`

Removes the Checklist Due Date from the Message item. Returns an error if the item is not on the Checklist.

`RetractAppointment([Boolean vSendCancelMessage])`

Requests appointment retraction from recipient mailboxes. The appointment's BoxType must be egwOutgoing. If the optional variant parameter vSendCancelMessage is set to VARIANT_FALSE, no cancellation message is sent to the recipients. If vSendCancelMessages is set to VARIANT_TRUE or the parameter is missing (that is, the VariantType is set to VT_ERROR or VT_EMPTY), a cancellation message is sent to all recipients.

Remarks

When an Appointment object is refreshed, it recursively refreshes its BusySearchResult object. It also updates the Message objects that are returned by its AutodateMessages property, but it does not recursively refresh the Message objects themselves.

StartDate

If StartDate is changed, EndDate is automatically adjusted to preserve the Duration. If EndDate is changed, Duration is automatically adjusted to preserve the StartDate. If Duration is changed, EndDate is automatically adjusted to preserve the StartDate.

The date is in the same format as Microsoft's Date structure (VT_DATE) and is represented as a double-precision number, where midnight on January 1, 1900 and January 2, 1900 is 2.0 and 3.0, respectively.

For more information on the Date structure, see Data Type Conversions (http://msdn.microsoft.com/library/default.asp?url=/library/en-us/com/htm/ctrans_8nhv.asp) and the VT_DATE description as part of VARIANT and VARIANTARG (http://msdn.microsoft.com/en-us/library/aa908601.aspx).

Visual Basic. For the StartDate property, you can use a date similar to the following examples:

```
10/29/2012 11:30:00 AM
2012/10/29 11:30:00 AM
#2012/10/29 11:30:00 AM#
```

You can also use military time (24-hour clock) and skip the a.m. and p.m. designation.

C++. For the StartDate property, you can fill a C SYSTEMTIME structure for your desired date. Then call the C SystemTimeToVariantTime function, which takes as input a pointer to your newly created SYSTEMTIME structure. The function returns a pointer to the DATE structure that can be used for the StartDate property.

Attachment

Describes an attachment to a Message object.

Properties

The following table lists properties along with their access and descriptions.

Property	Access	Description
Application	R/O	Application. The Application object.
AttachmentSize	R/O	Long. The size of the attachment (in bytes).
ContentId	R/O	String. The MIME Content-ID of the attachment.
ContentType	R/O	String. The MIME Content-Type of the attachment.
DisplayName	R/O	String. The descriptive name to be displayed to users.
FileName	R/O	String. The name of the file this attachment represents.
Hidden	R/O	Boolean. Whether or not the GroupWise client has hidden the attachment.
Message	R/O	Message. The Message object that this attachment represents.
ModifiedDate	R/O	Date. The date and time this attachment was last modified.
ObjType	R/O	Enum (AttachmentTypeConstants). The type of this attachment.
Parent	R/O	Attachments collection. The Attachments collection that owns this object.
Stream	R/O	Returns a GWStream object, which is used to retrieve data from large attachments. In the GWStream object, you can specify how many bytes of the attachment to read and also position the point in the attachment data to start reading. This functionality allows you to read the attachment data in several different data blocks. For an example of how to use this property, see the Example section below.

Methods

`Delete()`

Deletes the attachment from the associated attachments collection and the associated message. This method functions on draft or personal items only.

`Save(String Filename)`

Saves this attachment to the file specified by Filename. Not supported if ObjType is egwMessage.

Remarks

If ObjType is egwMessage, the Message property points to the attached message and the Filename property is ignored (it should be an empty string ("")). If ObjType is not egwMessage, the Filename property is the file name for the attached file, movie file, multimedia file, embedded OLE object, or linkedOLE object info. In this case, the Message property is ignored (it should be "Nothing").

An Attachment object is refreshed when its parent object is refreshed. When an Attachment object is refreshed, it recursively refreshes its Message object

Example

In GWStream, you can specify how many bytes of the attachment to read. You can also specify the starting point within the attachment data from which to start reading to read the attachment in chunked data blocks. The following example shows how to read data and write it to a file:

```
Private Sub btnSave_Click(ByVal sender As Object,
   ByVal e As System.EventArgs) Handles btnSave.Click

     Dim stream As GroupwareTypeLibrary.GWStream
     Dim arr() As Byte
     Dim str As String
     Dim fs As System.IO.FileStream
     Dim bw As System.IO.BinaryWriter

     On Error GoTo -1
     str = txtDirectory.Text + "\" + txtName.Text
     fs = New System.IO.FileStream
        (str, IO.FileMode.Create, IO.FileAccess.Write)
     bw = New System.IO.BinaryWriter(fs)

     stream = gwAttachment.stream
     arr = stream.Read(512)
     While arr.Length <> 0
          bw.Write(arr)
          If arr.Length <> 512 Then Exit While
          arr = stream.Read(512)
     End While
     fs.Close()
End Sub
```

The Seek() method specifies the position from which to start reading the data. By default, the position in the data is automatically updated when you read, so you don't need to call the Seek() method to read, unless you want to start reading from a different position. When a GWStream object is first retrieved, the read position is at the beginning of the data. You can call Seek() to reread data by moving the read position back to the beginning of the data.

The Write() method is only used to write data in a Stream object that can be passed in as the attachment data for the Attachments.AddEx()method.

Attachments

A collection of Attachment objects.

Properties

The following table lists properties along with their access and descriptions.

Property	Access	Description
Application	R/O	Application. The Application object.
Count	R/O	Long. The number of objects in this collection.
_NewEnum	R/O	Enumeration object. Implements IEnumVARIANT. For Windows only.
Parent	R/O	Message. The Message object that owns this collection.

Methods

`Attachment Add(String Filename, [AttachmentTypeConstants ObjType], [String DisplayName])`

Adds an existing message as an attachment to the message that owns this collection. The owning message's BoxType must be egwDraft or egwPersonal. Any other BoxType results in an exception being thrown. Creates a new Attachment object and adds it to the collection.

- The new attachment's Message property is set to Message
- Its ObjType property is set to egwMessage
- Its DisplayName property is set to DisplayName.

`Attachment Add(Message Message, [String DisplayName])`

Adds an existing file as an attachment to the message that owns this collection. The owning message's BoxType must be egwDraft or egwPersonal. Any other BoxType results in an exception being thrown. Creates a new Attachment object and adds it to the collection.

- The new attachment's Filename property is set to Filename
- Its ObjType property is set to ObjType (File is assumed if ObjType is omitted)
- Its DisplayName property is set to DisplayName. ObjType cannot be egwMessage.

`Attachment Item(VARIANT Index)`

DEFAULT. Returns the Attachment object specified by Index. Index may be a Long, a string, or a Message object. If Index is a Long, returns the Attachment object located at the given Index in the collection. Valid indexes are 1 through Count, inclusive. Throws an exception if the Index is outside of this range. If Index is a string, it represents the MessageID of the desired attachment. If Index is a Message object, it represents the desired Message object attachment.

`Attachment AddEx(GWStream Stream, [String DisplayName], [String ContentType], [String ContentId])`

Creates a new Attachment object and adds it to the collection. To use this version of the method, you must first get a GWStream object using the Stream() method. You write the attachment data into the GWStream object using the GWStream. Write() method.

- The Stream object must already be filled with the attachment data before the AddEx() method is called.

- The ObjType can only be File.
- Its DisplayedName property is set to DisplayName.
- ContentType is the MIME Content-Type of the attachment.
- ContentId is the MIME Content-ID of the attachment.

`Attachment AddEx(String Filename, [String DisplayName], [String ContentType], [String ContentId])`

Creates a new Attachment object and adds it to the collection.

- The new attachment's FileName property is set to Filename.
- The ObjType can only be File.
- Its DisplayName property is set to DisplayName. If the DisplayName is omitted, the Filename will be used for the DisplayName (minus the path portion).
- ContentType is the MIME Content-Type of the attachment.
- ContentId is the MIME Content-ID of the attachment.

`GWStream Stream()`

Creates a GWStreaming object. Once you have the object, you use the Write() method ti write the attachment data into the object. You can then pass the GWStream object into the AddEx() method.

Remarks

The Add method template is as follows:

`Add(VARIANT P1, [VARIANT P2], [VARIANT P3])`

The Add method checks the P1 type at runtime.

- If P1 is an string, the first form of Add is assumed; P2, if present, must be an ObjType Enum of type AttachmentTypeConstants, and P3 if present, must be a string.
- If P1 is a Message object, the second form of Add is assumed; P2 if present, must be a string, P3 must not be present.

An Attachments collection is refreshed when its parent object is refreshed. When an Attachments collection is refreshed, it recursively refreshes its contained Attachment objects.

Also see Annotate under Methods for the Message object, which adds an existing personal Note as an attachment.

In the past, the user of the ObjectAPI had no way to write a HTML message body (they always has to do a message body in plain text). For the GroupWise Client to understand the message body as being an HTML message body, the message body had to have certain characteristics. The HTML message body needed to be an attachment. The message body attachment name is required to be "text.htm". The message body has to be the MIME type (ContentType):"text/html". The AddEx method allows the user to specify those parameters.

It is also possible for the HTML message body to have embedded graphics. The text.htm file has references to those separate graphic files. The text.htm file uses the ContentId to reference the graphics. You then add the external graphic files as a separate attachment. You specify the ContentType and ContentId. The HTML message body is comprised of the text.htm attachment and any referenced graphics file attachment that have a ContentId. The client reads in the text.htm attachment and any following attachment until there is to dump out the attachments on an item that has an HTML message body using the ObjectAPI.

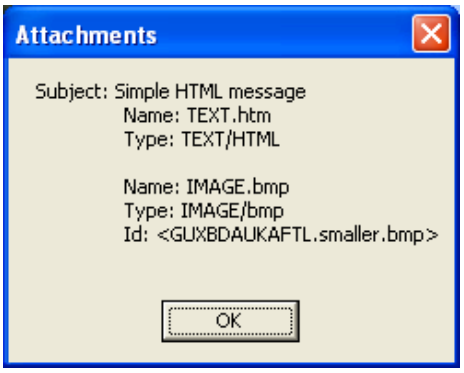
The AddEx method gives the user a way to stream in an attachment without writing the data to a file. For example, if the application wanted to format a HTML message body but not first write the data to a file, they can use the Stream method. You first get a stream (Gstream) object using the Stream() method. You write your attachment data to the stream using the GWStream. Write() method. You then pass the stream object in as the FileName parameter of the AddEx method.

Here is a sample HTML message body:

```
<HTML><HEAD>
<META http-equiv=Content-Type content="text/html; charset=utf-8">
<META content="MSHTML 6.00.6001.18148" name=GENERATOR></HEAD>
<BODY style="MARGIN:4px 4px 1px; FONT: 10pt Segoe UI">
<DIV><STRONG>This is a test.</STRONG></DIV>
<DIV><STRONG><IMG alt=""hspace=0 src="cid:GUXBDAUKAFTL.smaller.bmp" align=baseline
border=0></STRONG></DIV></BODY></HTML>
```

Here is output of a simple program to list the attachments:

Figure 3-1 *Attachments list*



Here is a code snippet that produces the message box:

```
Private Sub btnTest_Click()
   Dim gwAcct As Account9
   Dim gwApp As Application5
   Dim gwAttach As Attachment3
   Dim gwAttachs As Attachments2
   Dim gwMessage As Message3
   Dim gwMessages As Messages3
   Dim str As String
   str = "/ipa-" + txtServer.Text + " /ipp-" + txtPort.Text

   Set gwApp =CreatedObject("NovellGroupwareSession")
   Set gwAcct = gwApp.MultiLogin(txtUserId.Text, str, txtPassword.Text)
   Set gwFolder = gwAcct.MailBox
   Set gwMessages = gwFolder.Messages
   For Each gwMessage In gwMessages
      If gwMessage.Subject.PlainText = txtSubject.Text Then
         str = "Subject: " & txtSugject.Text Then
         Set gwAttachs = gwMessage.Attachments
         For Each gwAttach In gwAttachs
            str = str & chr(9) & "Name: " & gwAttach.DisplayName & chr(10)
            str = str & chr(9) & "Type: " & gwAttach.ContentType & chr(10)
            If "" <> gwAttachment.ContentId Then
               str = str & chr(9) & "Id: " & gwAttach.ContentId & chr(10)
```

```
            End If
            str = str & chr(10)
         Next
         MsgBox str
         Exit For
      End If
   Next
   Set gwAttach = Nothing
   Set gwAttachs = Nothing
   Set gwMessage = Nothing
   Set gwMessages = Nothing
   Set gwFolder = Nothing
   Set gwAcct = Nothing
   Set gwApp = Nothing
End Sub
```

BusySearchElement

Contains the busy search results for a recipient, or the combined results for all recipients.

Properties

The following table lists properties along with their access and descriptions.

Property	Access	Description
Address	R/O	Address. The address of the busy search recipient, address or nothing when this BusySearchElement contains a combined result.
Application	R/O	Application. The Application object.
Completed	R/O	Boolean. TRUE if the busy search has completed.
FreeBlocks	R/O	TimeBlocks collection. The collection of all free blocks of time.
Parent	R/O	Object. The BusySearchElements collection or BusySearchResult object that owns this object.
TargetType	R/O	Enum (AddressTargetTypeConstants). The target type (recipient type) of the user represented by Address, or undefined if this BusySearchElement contains a combined result.
TimeBlocks	R/O	TimeBlocks collection. The collection of all blocks of time (free, blocked, etc.).

Remarks

A BusySearchElement object is refreshed when its parent object is refreshed. When a BusySearchElement object is refreshed, it recursively refreshes its Address object and the TimeBlocks collections (returned by its FreeBlocks and TimeBlocks properties).

BusySearchElements

A collection of BusySearchElement objects.

Properties

The following table lists properties along with their access and descriptions.

Property	Access	Description
Application	R/O	Application. The Application object.
Count	R/O	Long. The number of objects in this collection.
_NewEnum	R/O	Enumeration object. Implements IEnumVARIANT. For Windows only.
Parent	R/O	BusySearchResult. The BusySearchResult object that owns this collection.

Methods

`BusySearchElement Item(Long Index)`

DEFAULT. Returns the BusySearchElement object located at the given Index in the collection. Valid indexes are 1 through Count inclusive. Throws an exception if the Index is outside of this range.

Remarks

A BusySearchElements collection is refreshed when its parent object is refreshed. When a BusySearchElements collection is refreshed, it recursively refreshes its contained BusySearchElement objects.

BusySearchResult

Provides partial status and results of an active busy search, and the final results of a completed busy search. A draft appointment creates a BusySearchResult object when the StartBusySearch method is called. This BusySearchResult object persists until a new search starts or until the appointment is no longer a draft because it is sent or canceled.

Properties

The following table lists properties along with their access and descriptions.

Property	Access	Description
Application	R/O	Application. The Application object.
BusySearchElements	R/O	BusySearchElements collection. Results (elements) for each recipient of the owning Appointment. Represents the partial status if Completed = FALSE, or the final result if Completed = TRUE.
CombinedResult	R/O	BusySearchElement. A combined snapshot result for all recipients. A blank time block indicates all recipients are unscheduled. Represents a snapshot of partial results if Completed = FALSE, or the final results if Completed = TRUE.
Completed	R/O	Boolean. TRUE if the search has finished.
Parent	R/O	Appointment. Appointment object that owns this object.
Range	R/O	Long. The number of days being searched.
StartDate	R/O	Date. The date on which the search was started.

Methods

`Refresh()`

Forces this BusySearchResult object to reread its property values from the message database. You can use this method to update the Completed property during a busy search. When a BusySearchResult object is refreshed, it recursively refreshes its BusySearchElements collection and the BusySearchElement object returned by its CombinedResult property

Remarks

To perform a busy search:

- Use Messages.Add in a folder to create a draft appointment.
- Fill in appointment Recipients and StartDate.
- Call StartBusySearch. A BusySearchResult object is created to contain the result.

Call Refresh to update the Completed property. When Completed is TRUE, the search is done and the BusySearchResult object contains the result.

Because the busy search can take some time, it executes asynchronously. While searching, the BusySearchResult object provides a snapshot of the partial results. The Completed property indicates when a search is finished.

Categories

Provides a category for a collection.

Properties

Property	Access	Description
Application	R/O	Application. The Application object.
Count	R/O	Long. The number of items in this collection.
_NewEnum	R/O	Enumeration object. Implements IEnumVARIANT. For Windows only.
Parent	R/O	IDispatch pointer. The Dispatch pointer at the object that owns this Collection.
ParentType	R/O	CategoryParentObjectTypeConstant. An enumeration value detailing what type of parent object owns this collection. Use this value to quickly determine how to cast the Parent IDispatch pointer returned in the Parent property.

Methods

`Category Add(CategoryDefinition NewCategory)`

Adds a new Category to the collection (see Section 4.11, “CategoryParentObjectTypeConstants,” on page 175). The new category is given by the passed in NewCategory Category Definition object. Returns a Category object. Throws an exception if the given Category Definition is already in the collection.

Category Item(Long Index)

Default. Returns the Category object located at the given Index in the collection. Valid indexes are 1 through Count, inclusive. Throws an exception if the index is outside of this range.

Category

Provides a category.

Properties

Property	Access	Description
Application	R/O	Application. The Application object.
Name	R/O	String. Returns the Name of this category, as given by the CategoryDefinition referenced by this object.
Parent	R/O	Categories. The Categories Collection that owns this object.
Primary	R/W	Boolean. Sets or retrieves whether this Category object is the Primary category for this collection. (Making a category "Primary" means the GroupWise Windows client uses this category as the value to sort on when displaying items in Category order. If the Category Definition for this Category contains a Color, a Primary designation displays this item in the Color given.)

Methods

`Delete()`

Removes this Category from the Mail, Appointment, Note, Task, Phone Message, Document Reference, or AddressBookEntry. If this Category was the Primary category, the object is left with no Primary category.

CategoryDefinition

Provides a category definition.

Properties

Property	Access	Description
Application	R/O	Application. The Application object.
Color	R/W	Long. The RGB value the Windows client uses to display an object that contains this category as the Primary category.
Name	R/W	String. Get or Put (Change) the name of this CategoryDefinition.
Parent	R/O	CategoryDefinitions. The CategoryDefinitions Collection that owns this object.

Methods

`Delete()`

Removes this CategoryDefinition from the database.

CategoryDefinitions

Provides a category definition for a collection.

Properties

Property	Access	Description
Application	R/O	Application. The Application object.
BackgroundColor	R/W	Long. RGB value to use in the background of this catagory.
CatagoryID	R/O	String. An ID that uniquely identifies this catagory definition.
Count	R/O	Long. The number of items in this collection.
ModifiedDate	R/O	Date. The date and time this catagory definition was last modified
_NewEnum	R/O	Enumeration object. Implements IEnumVARIANT. For Windows only.
Parent	R/O	Account. The Account object that owns this Collection.

Methods

`CategoryDefinition Add(String Name)`

Adds a Category Definition of the given name to the collection. Returns a CategoryDefinition object of the new Name. Throws an exception if the given Name is already in the collection.

`CategoryDefinition Item(Long Index)`

Default. Returns the CategoryDefinition object located at the given Index in the collection. Valid indexes are 1 through Count, inclusive. Throws an exception if the index is outside of this range.

Document

A document management document.

Properties

The following table lists properties along with their access and descriptions.

Property	Access	Description
AdditionalRights	R/O	DocumentAccessRightsCollection. Contains rights granted to specific users or groups. This collection will be empty if: (1) the user does not have rights to modify access rights (see the ModifySecurity property under the DocumentAccessRights object); or (2) GroupWise Remote is running. This is a GroupWise limitation because the ModifySecurity rights information is not known or guaranteed to be current when Remote is running. The user can perform any action on the document, but when those actions are uploaded, the synchronization is subject to the user's actual access rights.
Application	R/O	Application. The Application object.
Author	R/W	Address. The author of the document. During document creation, defaults to the same address as the Creator.
AuthorCreatorRights	R/O	DocumentAccessRights. Contains access rights to the document that are granted to the document's author or creator. This property is empty when running GroupWise Remote.
CreationDate	R/O	Date. The date and time this document was created.
Creator	R/O	Address. The creator of this document.
CurrentVersion	R/O	DocumentVersion. The latest revision of the document. All revisions later than the "official" version are also collectively referred to as "current."
DefaultRights	R/O	DocumentAccessRights. Contains access rights to the document that are granted to Everyone as a default. This property is empty when running GroupWise Remote.
DocumentLibrary	R/O	DocumentLibrary. The document library which contains this document.
DocumentNumber	R/O	Long. The number of this document. This number is unique within the document library.
DocumentType	R/W	DocumentType. The document type for this document. Categorizes this document according to specific document retention characteristics.
DocumentVersions	R/O	DocumentVersions collection. The collection of versions for this document.
Fields	R/O	Fields collection. The collection of user-defined fields for this document.
OfficialVersion	R/W	DocumentVersion. The version of this document deemed to be the approved or "official" version.
Parent	R/O	Documents collection. The Documents collection that owns this object.
Subject	R/W	String. The descriptive name, or subject, of the document.

Methods

`Delete()`

Deletes the document and all corresponding versions from the document library.

`DocumentVersion GetVersion(Long VersionNumber)`

Returns the document with the DocumentNumber equal to the specified VersionNumber.

`LocalDelete()`

Deletes this document only from the currently connected post office database. No synchronization will take place between the master database and a remote database. This method i

`Refresh()`

Forces this Document object and its associated DocumentRights and Fields objects to read values from the message database. The actual reading of a specific property may be postponed until the next time the property is accessed. This "lazy evaluation" is an optimization that avoids unnecessary reading of unaccessed properties.

`SetDefaults()`

Sets default fields in the document. Each document library defines a set of default fields for documents in the library. These defaults name particular field values that are normally set on each document (by default) in the library. Calling this method will set the field values on the document to their default values based on how the system administrator has configured the library.

Remarks

When a Document object is refreshed, it recursively refreshes its contained objects and collections.

DocumentAccessRights

Describes rights granted to individual users or groups.

Properties

The following table lists properties along with their access and descriptions.

Property	Access	Description
Application	R/O	Application. The Application object.
ModifySecurity	R/W	Boolean. TRUE if the user can modify the access rights of the owning document. This property cannot be set to TRUE if the RevokeAllRights property is TRUE.
Parent	R/O	DocumentAccessRightsCollection. The DocumentAccessRightsCollection that owns this object.
RevokeAllRights	R/W	Boolean. When set to TRUE, the user identified by the User property has all rights to the document revoked. When set to TRUE, the SecurityCurrentVersions, SecurityOfficialVersion, SecurityOtherVersions, and ModifySecurity properties have no meaning. Attempting to write to any of these properties while RevokeAllRights is TRUE will generate an exception. Also, if RevokeAllRights is set to TRUE, ModifySecurity is forced FALSE.
SecurityCurrentVersions	R/O	DocumentRights. Access rights for this document's current version. Current versions are defined as all document revisions newer than the official document version.
SecurityOfficialVersion	R/O	DocumentRights. Access rights for this document's official version.
SecurityOtherVersions	R/O	DocumentRights. Access rights for this document's unofficial, noncurrent versions (all document versions older than the official version.)
User	R/O	Address. Identifies the user or group for whom additional rights are being set. When the DocumentAccessRights object is obtained from a Document's AuthorCreatorRights or DefaultRights properties, the User property returns nothing.

Methods

`Delete()`

Deletes the DocumentAccessRights object, removing the object from the parent DocumentAccessRightsCollection.

Remarks

A DocumentAccessRights object is refreshed when its parent is refreshed. When a DocumentAccessRights object is refreshed, it recursively refreshes its contained objects.

DocumentAccessRightsCollection

A collection of DocumentAccessRights objects.

Properties

The following table lists properties along with their access and descriptions.

Property	Access	Description
Application	R/O	Application. The Application object.
Count	R/O	Long. The number of objects in this collection.
_NewEnum	R/O	Enumeration object. Implements IEnumVARIANT. For Windows only.
Parent	R/O	Document. The Document object that owns this collection.

Methods

`Add(VARIANT Address, Long Rights)`

If Address is an Address object, this method creates a new DocumentAccessRights object for that address. If Address is an Addresses collection, this method creates a new DocumentAccessRights object for each Address in the collection. Rights specifies the rights to be given. The value for Rights is derived from combining DocumentAccessRightsConstants with the bit-wise inclusive OR operator. Because there is typically no need to access the new DocumentRights object immediately after creation, this Add method does not return the newly created object. This method will fail if the user does not have rights to modify the access rights of the owning document.

`DocumentAccessRights Item(Long Index)`

DEFAULT. Returns the DocumentAccessRights object located at the given Index in the collection. Valid indexes are 1 through Count inclusive. Throws an exception if the Index is outside of this range.

Remarks

A DocumentAccessRights collection is refreshed when its parent object is refreshed. When a DocumentAccessRights collection is refreshed, it recursively refreshes its contained DocumentAccessRights objects.

DocumentIterator

Iterates through Documents collections, successively returning each Document object. Multiple DocumentIterator objects can be active on the same collection. Each iterator maintains its own internal position and does not affect the state of the Documents collection over which it is iterating.

Properties

The following table lists properties along with their access and descriptions.

Property	Access	Description
Application	R/O	Application. The Application object.
Parent	R/O	Documents collection. The Documents collection that owns this object.

Methods

`DocumentIterator Clone()`

Creates an independent DocumentIterator instance. The clone starts at the position last access by the original iterator. Use Clone() to save a position before calling Skip or Reset.

`Document Next()`

Returns the next document in a collection, or nothing after the last document.

`Reset()`

Resets iteration to the collection beginning so that Next() returns the first document.

`Skip(Long NumDocuments)`

Skips the next NumDocuments documents. Skip does nothing if there are not NumDocuments documents left to skip, or if NumDocuments < 0.

Remarks

Both DocumentIterator and IEnumVARIANT (obtained from the Documents collection's _NewEnum method) provide Documents collection iteration. DocumentIterator objects are accessible in all development environments while IEnumVARIANT is a COM interface inaccessible in environments such as Delphi. The usual alternative to IEnumVARIANT is:

```
for i:= 1 to Count
```

This does not work with large collections, such as Documents, which do not support Count or Item.

The DocumentIterator object does not support refresh.

DocumentLibraries

A collection of DocumentLibrary objects.

Properties

The following table lists properties along with their access and descriptions.

Property	Access	Description
Application	R/O	Application. The Application object.
Count	R/O	Long. The number of objects in this collection.
_NewEnum	R/O	Enumeration object. Implements IEnumVARIANT. For Windows only.
Parent	R/O	Account. The Account object that owns this collection.

Methods

`DocumentLibrary Item(VARIANT Index)`

DEFAULT. Returns the DocumentLibrary object specified by Index. Index may be a Long or a string. If Index is a Long, returns the DocumentLibrary object located at the given Index in the collection. Valid indexes are 1 through Count, inclusive. Throws an exception if the Index is outside of this range. If Index is a string, returns the DocumentLibrary object whose LibraryID is equal to Index.

Remarks

A DocumentLibraries collection is refreshed when its parent object is refreshed. When a DocumentLibraries collection is refreshed, it updates the DocumentLibrary objects which are in the collection but it does not recursively refresh the DocumentLibrary objects themselves.

DocumentLibrary

A document management library.

Properties

The following table lists properties along with their access and descriptions.

Property	Access	Description
Application	R/O	Application. The Application object.
CurrentArchiveBytes	R/O	Long. The number of bytes that the current archive location uses on disk.
CurrentArchiveLocation	R/O	String. The path to the library's archive.
Description	R/O	String. The description of this library.
Documents	R/O	Documents collection. The collection of documents in this library.
DocumentTypes	R/O	DocumentTypes collection. The collection of document types in this library.
FieldDefinitions	R/O	FieldDefinitions collection. The collection of field definitions for this library.
LibraryID	R/O	String. The unique ID for this library.
MaxArchiveBytes	R/O	Long. The maximum number of bytes that can be archived before a new archive location will be created.
Name	R/O	String. The descriptive display name of this library.
NextDocumentNumber	R/O	Long. The number that will be assigned to the next document added to this library.
Parent	R/O	DocumentLibraries collection. The DocumentLibraries collection that owns this object.
StartingVersionNumber	R/O	Long. The default starting version number for new document versions created in this library.

Methods

`Document GetDocument(Long DocumentNumber)`

Returns the Document object with the specified DocumentNumber.

`DocumentVersionEvents GetDocumentVersionEvents(Long DocNumber)`

Returns the DocumentVersionEvents collection pertaining to the document whose DocumentNumber is equal to DocNumber.

`IncrementArchiveLocation()`

GroupWise archives documents according to a progressive directory naming scheme. Calling this method will cause the document library to begin archiving documents in the next available directory.

This method is not available in GroupWise Remote.

`Refresh()`

Forces this DocumentLibrary object and its associated object and collections to reread property values from the message database. The actual reading of a specific property may be postponed until the next time the property is accessed. This "lazy evaluation" is an optimization that avoids unnecessary reading of unaccessed properties.

DocumentReference

Refers to a specific version of a document management document. A subtype of Message.

Properties

The following table lists properties along with their access and descriptions.

Property	Access	Description
Archived	R/W	Boolean. TRUE if this message has been archived.
Categories	R/O	Categories Collection. The collection of Category objects on this Mail, Appointment, Document Reference, Note, Phone Message, Task, or AddressBookEntry.
ChecklistCompleted	R/W	Boolean. TRUE if the item is part of the Checklist folder Messages collection or a Task item AND the item has been marked Completed. FALSE otherwise.
ChecklistCompletedDate	R/W	Date. The date this item was marked completed.
ChecklistDueDate	R/W	Date. The date this item should be Completed. Returns an error if the Message item is not a Task or is not part of the Checklist folder Messages collection.
ChecklistPercentComplete	R/W	Integer. Between 0 and 100 percent, how complete the checklist item is. If this value is set to a number outside the 0 to 100 range, an error is returned. If the value is set to 100, the item will also be marked complete.
ChecklistShowOnCalendarDate	R/W	Date. The date this item should appear on the calendar task list.
CommonMessageID	R/O	String. The ID of the message, regardless of its location. For example, an email will have the same CommonMessageID for both Sent and Received.
DeliveredDate	R/O	Date. The date and time this message item was delivered to the account.
Document	R/O	Document. The original document from which the document version and this document reference have been generated.
DocumentLibrary	R/O	DocumentLibrary. The document library in which the document version resides.
DocumentSize	R/O	Long. Returns the size of the document that is pointed to by this reference.
DocumentVersion	R/O	DocumentVersion. The document version to which this document reference refers.
Forwarded	R/O	Boolean. TRUE if this item has been forwarded to another user. Otherwise, FALSE.
MessageProperty	R/O	String. Based on the MessagePropertiesConstant passed in, a different string is returned.
PersonalSubject	R/W	String. A text string to be shown by the GroupWise Windows client as the Subject text. This text appears only for the current user. The normal subject or their own Personal Subject text appears for other users.

Property	Access	Description
RefType	R/O	Enum (DocumentReferenceTypeConstants). The type of this document reference.
Replied	R/O	Boolean. TRUE if this item has been replied to. Otherwise, FALSE.
Sharer	R/O	Address. The user who added this message to a shared folder. If this message was not added to a shared folder, this will be the same as the Sender property.
ShowOnChecklist	R/W	Boolean. TRUE if the item is part of the Checklist folder Messages collection. FALSE otherwise.
Size	R/O	Returns the size of the attachments and message body. If there are no attachments, only the size of the message body is returned.

Methods

See Message.

`LocalDelete()`

Deletes this message only from the currently connected post office database. No synchronization will take place between the master database and a remote database.

`RemoveChecklistDueDate()`

Removes the Checklist Due Date from the Message item. Returns an error if the item is not on the Checklist.

Remarks

DocumentReference objects are typically personal items (read/write objects). However, when a DocumentReference is attached (encapsulated) to another message, the encapsulation is ReadOnly.

DocumentRights

Details a set of access rights associated with a document. The object is typically used to list additional rights that are added to a user's base level access. The additional rights are associated with a specific set of document versions (such as the "current" version).

Properties

The following table lists properties along with their access and descriptions.

Property	Access	Description
AllowDelete	R/W	Boolean. TRUE if the user can delete the document version.
AllowEdit	R/W	Boolean. TRUE if the user can modify the document version.
AllowShare	R/W	Boolean. TRUE if the user can share the document version with other users. Document versions can be shared by adding them to shared folders, and by forwarding them as attachments to messages.
AllowView	R/W	Boolean. TRUE if the user can view the document version.
Application	R/O	Application. The Application object.
BitMask	R/W	Long. A bit mask containing one bit for each Boolean property. See DocumentAccessRightsCollection for the bit definitions. This property is synchronized with the Boolean properties.
Parent	R/O	DocumentAccessRights. The DocumentAccessRights object that owns this object.

Remarks

The properties of the DocumentRights object are writable only by users who have been granted the ModifySecurity right (in the DocumentAccessRights object).

A DocumentRights object is refreshed when its parent object is refreshed.

Documents

A collection of Document objects contained in a document library.

Properties

The following table lists properties along with their access and descriptions.

Property	Access	Description
Application	R/O	Application. The Application object.
_NewEnum	R/O	Enumeration object. Implements IEnumVARIANT.For Windows only.
Parent	R/O	DocumentLibrary. The DocumentLibrary object that owns this collection.

Methods

`Document Add(String Filename, [VARIANT DocumentType])`

Creates a new document and adds it to the collection. It does not directly support the creation of a new document using a template document. To locate a template document, call the DocumentVersion object's CopyOut method, then use the resulting file with the current Add method.

Filename is the file from which the document is created.

DocumentType specifies the new documents DocumentType. If DocumentType is a String, it specifies the DocumentType object's name. If DocumentType is an existing DocumentType object, the new document will be of that DocumentType.

When you add a document in GroupWise Remote, the new document is assigned a temporary document and Open Document Management API (ODMA) identifier. When the remote account is synchronized with the master account, the ID will change. DocumentReference objects that refer to the temporary ID will be modified. Using a DocumentReference, an Independent Software Vendor (ISV) can track the document through a synchronization cycle.

If you pass a string as the DocumentType parameter, the string must be a valid document type. To figure out the set of valid document types, use Object API and ask for the DocumentTypes (page 83) from a DocumentLibrary (page 75) object. You can then iterate through this collection to find the list of valid DocumentTypes.

If a user must enter a Document Type to a third-party application, fill in a combo box, list box, or other window control with the DocumentType Names from that DocumentTypes collection. Use the Windows control selection as the Document Type parameter to the Documents::Add method.

In addition, you may add a document in GroupWise Remote, but you may not have rights on the master system to add the document there. If this is the case, the document will exist in your local copy of the document library, but it will not be added to the master system. In this case, failure to upload a new document will not delete the document from the local (remote) library. This condition cannot be detected through the Object API for GroupWise.

`Document AddEx(String Filename, [VARIANT DocType], [VARIANT DocRefFolder])`

Creates a new document and adds it to the collection. The first two parameters are the same as the parameters for the Add method. DocRefFolder, if supplied, must be a Folder object. It represents the folder in which you wish to add a document reference to the document you are adding to the collection.

`DocumentIterator CreateDocumentIterator()`

Creates a new DocumentIterator object for iterating over the Documents in this collection.

Remarks

Because a Documents collection can contain several thousand documents, the collection is designated a "large collection." This means that it does not support a Count property or an Item method. Instead, it provides access to its elements through an iterator object. In addition, the Documents collection parent (DocumentLibrary) provides the NextDocumentNumber property and the GetDocument method to allow access to documents by document number. GetDocument, however, throws an exception for document numbers less than NextDocumentNumber because the document may have been deleted (document numbers are not reused) or the user may not have access rights to the document.

A Documents collection is refreshed when its parent object is refreshed. When a Documents collection is refreshed, it updates the documents in the collection but does not recursively refresh the Document objects themselves.

DocumentType

A document management document type.

Properties

The following table lists properties along with their access and descriptions.

Property	Access	Description
Application	R/O	Application. The Application object.
CollectionParent	R/O	The DocumentTypes collection that owns this object.
DisposalMethod	R/O	Enum (DisposalMethodConstants) Identifies the method for disposing of the document.
DocumentLife	R/O	Long. The number of days, after the last modification to the document, after which the document will be "disposed of" by the disposal method specified in the DisposalMethod property.
MaxVersionCount	R/O	Long. The maximum number of versions for any documents of this document type.
Name	R/O	DEFAULT. String. The name of this document type.
Parent	R/O	The DocumentLibrary that owns this object.

Remarks

A DocumentType object is refreshed when its parent object is refreshed.

DocumentTypes

A collection of DocumentType objects.

Properties

The following table lists properties along with their access and descriptions.

Property	Access	Description
Application	R/O	Application. The Application object.
Count	R/O	Long. The number of objects in this collection.
_NewEnum	R/O	Enumeration object. Implements IEnumVARIANT. For Windows only.
Parent	R/O	DocumentLibrary. The DocumentLibrary object that owns this collection.

Methods

`DocumentType Item(VARIANT Index)`

DEFAULT. Returns the DocumentType object specified by Index. Index may be a Long or a string. If Index is a Long, returns the DocumentType object located at the given Index in the collection. Valid indexes are 1 through Count, inclusive. Throws an exception if the Index is outside of this range. If Index is a string, returns the DocumentType object whose Name is equal to Index.

DocumentVersion

A document management document version.

Properties

The following table lists properties along with their access and descriptions.

Property	Access	Description
Application	R/O	Application. The Application object.
Archived	R/O	Boolean. TRUE if this document version is archived.
CheckedOut	R/O	Boolean. TRUE if this document version is checked out.
ConnectedReadWrite	R/O	Boolean. TRUE if this document version has been checked out from the master account and downloaded.
CreationDate	R/O	Date. The date and time this document version was created.
Creator	R/O	Address. The creator of this document version.
Description	R/W	String. The description of this document version.
Document	R/O	Document. The document from which this document version originated.
DocumentLibrary	R/O	DocumentLibrary. The document library in which this document version is stored.
DocumentVersionEvents	R/O	DocumentVersionEvents collection. Events that have affected this document version.
ExpirationDate	R/O	Date. The date and time this document version will expire.
InUse	R/O	Boolean. TRUE if this document version has been retrieved by an application.
ODMADocumentID	R/O	String. Document ID for Open Document Management API (ODMA). When adding a document in GroupWise Remote, the new document is assigned a temporary ODMA identifier. When the remote account is synchronized with the master account, the ID will change. However, DocumentReference objects that refer to the temporary ID will be fixed. Using a DocumentReference, an Independent Software Vendor (ISV) can then track the document through a synchronization cycle.
OriginalFileType	R/W	String. The external file extension for this document.
Parent	R/O	DocumentVersion collection. The DocumentVersions collection that owns this object.
RetrievalDate	R/O	Date. The date and time this document version was last retrieved by an application.
Retriever	R/O	Address. The user who last retrieved this document version.
StagedFilename	R/O	String. The full path and file name where the document version was last staged (stored on disk when retrieved by an application). If the document has not been staged, this property returns an empty string ("").
VersionNumber	R/O	Long. The version number of this document version. This number is unique within the document versions for each document.

Methods

`CheckIn([String Filename], [CheckInConstants StatusChange])`

Checks in the version. Throws an exception if the version is not checked out. Filename is the file containing the new content for the version. If Filename is omitted, the version content is not changed. StatusChange indicates whether to change the versions status to checked in (CheckIn, the default) or leave it checked out (LeaveCheckedOut). Passing LeaveCheckedOut is equivalent to checking the document in, then checking it out again. If StatusChange is omitted, it defaults to egwDoCheckIn. To check in a file, the Filename parameter is required. The name of the file to check in can be obtained from the StagedFilename property. See CheckInConstants.

`CheckOut([String Filename])`

Checks out the version. Throws an exception if the version is already checked out. Filename is the file to which the version content is copied. If Filename is omitted, the version content can be obtained later with the CopyOut method. Must eventually be matched by a CheckIn.

`CopyOut(String Filename)`

Copies the version content to a file even if the version is checked out or retrieved.

`Delete()`

Deletes this document version from the document library. If this is the last document version, this method will also delete the original document.

`EndPreview([String Filename])`

Ends a Preview. Throws an exception if the version is not in a preview state (detected by the document management engine). Calling this method will log an end preview event for the document, and delete the file created by the Preview method. If Filename is omitted, this method will try to use the StagedFilename. If the StagedFilename is an empty string (""), this method will do nothing.

`EndRetrieve([String Filename], [EndRetrieveStatus StatusChange])`

Ends a version retrieval. Throws an exception if the version is not retrieved. Filename is the file containing the new content for the version. If Filename is omitted, the version's content is not changed. StatusChange indicates whether to change the versions status to not in use (the default) or leave it in use. Passing egwLeaveInUse is equivalent to calling EndRetrieve followed by Retrieve. If all parameters are omitted, this method will end retrieval of the file specified by the StagedFilename. If the StagedFilename is an empty string (""), this method will do nothing.

`Preview([String Filename])`

Copies the version content to a file even if the version is checked out or retrieved.

`Refresh()`

Forces this DocumentVersion object, its associated Address objects returned by its Creator and Retriever properties, and its DocumentVersionEvents collection to reread property values from the message database. The actual reading of a specific property may be postponed until the next time the property is accessed. This "lazy evaluation" is an optimization that avoids unnecessary reading of unaccessed properties.

`RemoteEndRetrieve([VARIANT Filename], [EndRetrieveStatus Statuschange])`

Ends retrieval of a previously retrieved version on a Remote system.

`Retrieve([String Filename])`

Retrieves the document version to a file specified by Filename. Throws an exception if the version is already retrieved. Must eventually be matched by an EndRetrieve.

Remarks

CheckOut, CheckIn, and CopyOut are identical to Retrieve, EndRetrieve, and Preview because they all copy the version content to a file accessible by other programs. However, CheckOut, CheckIn, and CopyOut grant users file access to transfer the file, modify it using applications, and so forth, while Retrieve, EndRetrieve, and Preview open the file in an application, and cease access when the application closes the file. The distinction is important to a document history because each method logs varying DocumentVersionEvents.

DocumentVersionEvent

A document management version event.

Properties

The following table lists properties along with their access and descriptions.

Property	Access	Description
Application	R/O	Application. The Application object.
CreationDate	R/O	Date. The date this event was created.
Document	R/O	Document. The document which originated the document version to which this event occurred.
DocumentLibrary	R/O	DocumentLibrary. The document library where the document version is stored.
DocumentVersion	R/O	DocumentVersion. The document version on which this event occurred.
EventType	R/O	DEFAULT. Enum (EventTypeConstants). The type of event this DocumentVersionEvent object represents.
Filename	R/O	String. The path and file name where the content of the document version is stored.
Parent	R/O	DocumentVersionEvents collection. The DocumentVersionEvents collection that owns this object
User	R/O	Address. The user who performed this event.

Remarks

A DocumentVersionEvent object is refreshed when its parent object is refreshed. It recursively refreshes the Address object returned by its User property.

DocumentVersionEvents

A collection of DocumentVersionEvent objects.

Properties

The following table lists properties along with their access and descriptions.

Property	Access	Description
Application	R/O	Application. The Application object.
Count	R/O	Long. The number of objects in this collection.
_NewEnum	R/O	Enumeration object. Implements IEnumVARIANT. For Windows only.
Parent	R/O	DocumentVersion object. The DocumentVersion object that owns this collection.

Methods

`DocumentVersionEvent Item(VARIANT Index)`

DEFAULT. Returns the DocumentVersionEvent object located at the given Index in the collection. Index may be a Long, or a string that represents a Long. Valid indexes are 1 through Count inclusive. Throws an exception if the Index is outside of this range.

Remarks

A DocumentVersionEvents collection is refreshed when its parent object is refreshed. When a DocumentVersionEvents collection is refreshed, it recursively refreshes its contained DocumentVersionEvent objects.

DocumentVersions

A collection of DocumentVersion objects.

Properties

The following table lists properties along with their access and descriptions.

Property	Access	Description
Application	R/O	Application. The Application object.
Count	R/O	Long. The number of objects in this collection.
_NewEnum	R/O	Enumeration object. Implements IEnumVARIANT. For Windows only.
Parent	R/O	Document object. The Document object that owns this collection.

Methods

`DocumentVersion Add()`

Creates a new version for the document represented by the DocumentVersions collection. The new version is returned. Creating a new version copies the document's information (file) for the new version. Modifying the new version will not change any previous versions.

`DocumentVersion Item(VARIANT Index)`

DEFAULT. Returns the DocumentVersion object located at the given Index in the collection. Index may be a Long, or a string that represents a Long. Valid indexes are 1 through Count inclusive. Throws an exception if the Index is outside of this range.

Remarks

A DocumentVersions collection is refreshed when its parent object is refreshed. When a DocumentVersions collection is refreshed, it updates the DocumentVersion objects in the collection, but it does not recursively refresh the DocumentVersion objects themselves.

DownloadStatus

Provides information about a message. Indicates which parts of the message have been downloaded and are available for read/write operations.

Properties

The following table lists properties along with their access and descriptions.

Property	Access	Description
AttachmentsState	R/O	Enum (DownloadStateConstants). The status of the message attachments.
BodyTextState	R/O	Enum (DownloadStateConstants). The status of the message body text.
RecipientsState	R/O	Enum (DownloadStateConstants). The status of the message recipients

DraftAutoDates

Provides a date.

Properties

Property	Access	Description
Application	R/O	Application. The Application object.
Count	R/O	Long. The number of items in this collection.
_NewEnum	R/O	Enumeration object. Implements IEnumVARIANT. For Windows only.
Parent	R/O	Object. The Appointment, Note, or Task object that owns this Collection.

Methods

`Add(Date newAutoDate)`

Adds a Date to the collection. Throws an exception if the given newAutoDate occurs before this day.

`DATE Item(Long Index)`

Default. Returns a Date value located at the given Index in the collection. Valid indexes are 1 through Count, inclusive. Throws an exception if the index is outside of this range.

`Remove(Long Index)`

Removes a Date value located at the given Index from the collection. Throws an exception if the index is outside the range of 1 through Count, inclusive.

EMailAddress

Provides an email address.

Properties

Property	Access	Description
Application	R/O	Application. The Application object.
MailAddress	R/O	String. Returns the Email Address text of this object.
Parent	R/O	AddressBookEntry. Returns the Address Book entry that owns this object.
DefaultAddress	R/O	Boolean. Is this Email Address the "default" address for this collection? (Making an Email Address the default means the GroupWise system will use this Email address whenever the parent Address Book entry is added to a message for Sending, Forwarding, etc.)

Methods

`Delete()`

Removes this Email Address from the parent Address Book Entry.

`MakeDefaultAddress()`

Sets this address as the Default Email Address. (There must be a default address for each collection. Therefore, setting an Email Address as the default was coded as a method, not a Property that can have FALSE passed in.)

EMailAddresses

Provides an email address.

Properties

Property	Access	Description
Application	R/O	Application. The Application object.
Count	R/O	Long. The number of items in this collection.
_NewEnum	R/O	Enumeration object. Implements IEnumVARIANT. For Windows only.
Parent	R/O	AddressBookEntry pointer. The Dispatch pointer at the Address Book Entry object that owns this Collection.

Methods

`EMailAddress Add(String NewEMailAddress)`

Adds a new Email Address to the collection. The new address is given by the passed in String NewEMailAddress. Returns an EMailAddress object. Throws an exception if the given Email Address is already in the collection.

`EMailAddress Item(Long Index)`

Default. Returns the EMailAddress object located at the given Index in the collection. Valid indexes are 1 through Count, inclusive. Throws an exception if the index is outside of this range.

Field

Represents a user-defined field in an AddressBookEntry, Document, or Message. In an AddressBookEntry or Document, it can also represent a predefined field.

Properties

The following table lists properties along with their access and descriptions.

Property	Access	Description
Application	R/O	Application. The Application object.
FieldDefinition	R/O	FieldDefinition. Defines the restrictions for this field.
FieldID	R/O	Long. Field identifier.
Name	R/O	String. The name of this Field. This property is case-sensitive and must match the Name property of an existing FieldDefinition object.
Parent	R/O	Fields collection. The Fields collection that owns this object.
Value	R/W	DEFAULT. VARIANT. The value of this Field. Can be a string, numeric, date, or binary value.

Methods

`Delete()`

Deletes a field from an AddressBookEntry, Document, or Message. Also deletes the field from its parent Fields collection. If you delete a field, the deletion will not be synchronized across a remote link.

`Boolean Validate(VARIANT Value)`

Returns TRUE if the given Value is a valid value for this field.

Remarks

A Field object is refreshed when its parent object is refreshed.

FieldDefinition

Defines a field for an Account, AddressBook, or DocumentLibrary.

Properties

The following table lists properties along with their access and descriptions.

Property	Access	Description
Application	R/O	Application. The Application object.
FieldID	R/O	Long. Field identifier.
FieldType	R/O	Enum (FieldTypeConstants). The type of field this FieldDefinition object defines. The only valid FieldType for a Field object in an AddressBook is String.
HasLookupTable	R/O	Boolean. TRUE if the field has a lookup table associated with it.
Hidden	R/O	Boolean. TRUE if the field is hidden. If the field is hidden, it will not show up in any User Interface (UI).
MaximumLength	R/O	Long. The maximum string length of the field.
MaximumValue	R/O	Long. The maximum numerical value the field can have.
MinimumValue	R/O	Long. The minimum numerical value the field can have.
Name	R/O	DEFAULT. String. The name of the field being defined. This property is case-sensitive and must match an existing Name property in the Field object.
Parent	R/O	FieldDefinitions collection. The FieldDefinitions collection that owns this object.
ReadOnly	R/O	Boolean. TRUE if the field is read-only (i.e. the field's Value property cannot be changed).
RelatedFieldDefinition	R/O	FieldDefinition. A FieldDefinition object related to this one which constrains the values this field definition can have.
Required	R/O	Boolean. TRUE if the field is a required field (i.e. a value must be assigned to the field's Value property).
StringCase	R/O	Enum (FieldDefinitionCase). The string case of the field.

Methods

`Delete()`

Deletes this field definition from the parent collection or deletes a field definition that is associated with the Account (page 26) or DocumentLibrary (page 75) objects. It will not delete a field definition that is associated with the AddressBook (page 37) object.

`LookupTableEntries GetLookupTable([VARIANT Value])`

Returns the lookup table entries associated with this field definition and restricted by the given Value. Value can be either a string or a Field object. If Value is a string, returns the lookup table entries, restricted by Value. If Value is a Field object, returns the lookup table entries, restricted by the field's Value property.

Remarks

A FieldDefinition object is refreshed when its parent is refreshed.

FieldDefinitions

Contains the FieldDefinition objects for an Account, AddressBook, or DocumentLibrary. The root Account and each AddressBook have independent collections. The root Account collection includes user-defined fields that can appear in its messages. An AddressBook collection includes user-defined and predefined fields that can appear in its entries.

Properties

The following table lists properties along with their access and descriptions.

Property	Access	Description
Application	R/O	Application. The Application object.
Count	R/O	Long. The number of objects in this collection.
_NewEnum	R/O	Enumeration object. Implements IEnumVARIANT. For Windows only.
Parent	R/O	Object. The Account, AddressBook, or DocumentLibrary object that owns this collection.

Methods

`FieldDefinition Add(String Name, FieldTypeConstants FieldType)`

Defines a field with the given Name and FieldType. See FieldTypeConstants. Throws an exception when the owning Account or AddressBook has a FieldDefinition object that includes the same Name and FieldType. You cannot add FieldDefinition objects to a document library with this method.

`FieldDefinition Item(String FieldName, FieldTypeConstants FieldType)`

DEFAULT. Returns the FieldDefinition object with the given of FieldName and FieldType. See FieldTypeConstants.

`FieldDefinition Item(Long Index)`

DEFAULT. Returns the FieldDefinition object located at the given Index in the collection. Valid indexes are 1 through Count. Throws an exception when Index is outside of this range.

Remarks

The Item method template is as follows:

`Item(VARIANT P1, [VARIANT P2]).`

The Item method checks the P1 type at runtime.

- If P1 is an string, the first form of Item is assumed; P2 must be a FieldType of type FieldTypeConstants.
- If P1 is a Long, the second form of Item is assumed; P2 is ignored.

A FieldDefinitions collection is refreshed when its parent object is refreshed. When a FieldDefinitions collection is refreshed, it recursively refreshes its contained FieldDefinition objects.

Fields

Contains the Field objects for an AddressBookEntry, Document, or Message object.

Properties

The following table lists properties along with their access and descriptions.

Property	Access	Description
Application	R/O	Application. The Application object.
Count	R/O	Long. The number of objects in this collection.
_NewEnum	R/O	Enumeration object. Implements IEnumVARIANT. For Windows only.
Parent	R/O	Object. The Account, AddressBookEntry, or Document object that owns this collection.

Methods

`Field Add(String Name, FieldTypeConstants FieldType, VARIANT Value)`

Adds a Field with the given Name, FieldType (see FieldTypeConstants), and Value to the collection. Throws an exception if a FieldDefinition with the same Name and FieldType does not exist.

`Field Item(String FieldName, FieldTypeConstants FieldType)`

DEFAULT. Returns the Field object with the given FieldName and FieldType. See FieldTypeConstants. FieldName must be a string.

`Field Item(Long Index)`

DEFAULT. Returns the Field object located at the given Index in the collection. Valid indexes are 1 through Count inclusive. Throws an exception when Index is outside of this range.

Remarks

The Item method template is as follows:

```
Item(VARIANT P1, [VARIANT P2]).
```

The Item method checks the P1 type at runtime.

- If P1 is an string, the first form of Item is assumed; P2 must be a FieldType of type FieldTypeConstants.
- If P1 is a Long, the second form of Item is assumed; P2 is ignored.

A Fields collection is refreshed when its parent object is refreshed.

Filter

Represents saved filter information.

Properties

The following table lists properties along with their access and descriptions.

Property	Access	Description
Application	R/O	Application. The Application object.
Description	R/W	String. The description of this filter.
Expression	R/W	DEFAULT. String. The saved filter expression. The syntax of this property should match that used in the Filter dialog box of the GroupWise client.
LastAccessedDate	R/O	Date. The last date the filter was accessed. This date affects the most recent filters processing in the GroupWise client. The date can be updated by calling the TouchAccessedDate method.
Name	R/W	String. The name of this filter.
Parent	R/O	Filters collection. The Filters collection that owns this object.

Methods

`Delete()`

Deletes this filter from its parent collection.

`TouchAccessedDate()`

Updates the LastAccessedDate to the current date and time.

Remarks

A Filter object is refreshed when its parent object is refreshed. When a Filter object is refreshed, it rereads property values from the message database.

Filters

A collection of Filter objects that represent the saved filters associated with an account.

Properties

The following table lists properties along with their access and descriptions.

Property	Access	Description
Application	R/O	Application. The Application object.
Count	R/O	Long. The number of objects in this collection.
_NewEnum	R/O	Enumeration object. Implements IEnumVARIANT. For Windows only.
Parent	R/O	Account object. The Account object that owns this collection.

Methods

`Filter Add(String Name, String Expression)`

Creates a new saved filter with the given Name and Expression. The description of the filter can be set by accessing the returned Filter object.

`Filter Item(VARIANT Index)`

DEFAULT. Returns the Filter object specified by Index. Index may be a Long or a string. If Index is a Long, returns the Filter object located at the given Index in the collection. Valid indexes are 1 through Count, inclusive. Throws an exception if the Index is outside of this range. If Index is a string, returns the Filter object whose Name is equal to Index.

`Refresh()`

Forces this Filters collection to recursively refresh its contained Filter objects.

Folder

A GroupWise message store folder.

Properties

The following table lists properties along with their access and descriptions.

Property	Access	Description
AllFolders	R/O	Folders collection. All folders directly or indirectly contained by this folder.
Application	R/O	Application. The Application object.
CalendarColor	R/W	Long. Of this is not the calendar folder or a sub calendar folder, an error will be returned. The RGB value that items in this calendar will be displayed in.
ContactsAddressBook	R/W	AddressBook. Returns or sets the Address Book associated with a Contacts folder. Note: This property is valid only on the Contacts folder. All other folders result in an error.
Description	R/W	String. The description of this folder.
Fields	R/O	Fields collection. The collection of user-defined fields for this folder.
FolderID	R/O	String. Unique folder ID. Persistent across sessions.
FolderRights	R/O	FolderRightsCollection. Access rights given to specific users.
Folders	R/O	Folders collection. The collection of sub-folders to this folder.
Messages	R/O	Messages collection. The collection of messages in this folder. Note: A Contacts folder is a shortcut to an Address Book. Calling for the Messages collection on the Contacts folder results in an error.
ModifiedDate	R/O	Date. The date and time this folder was last modified.
Name	R/W	String. The name of this folder. The Name for the root folder is the User's Display Name, concatenated with the localized word for “Home.” For example, the root folder name for user John Doe is “John Doe Home.” Your application might need to check the ObjType property of a folder to ensure that it is the true root folder. If it is the root folder, you can access the Owner property of the account, then retrieve the DisplayName of the user from the returned Address object (which is the same string that is used to name the root folder in previous GroupWise versions).
ObjType	R/O	Enum (FolderTypeConstants). The type of this folder.
Owner	R/O	Address. The user who owns this folder.
Parent	R/O	Object. The Folders collection that owns this folder, or if this is the root folder, the Account object that owns this Folder.
ParentFolder	R/O	Folder. The parent folder to this folder, or "Nothing" if this is the root folder.

Property	Access	Description
Position	R/W	Long. The sequence number or position of this folder inthe folder list. **NOTE:** All other sibiling folders will be resequenced automatically.
PublishCalendar	R/W	Boolean. TRUE if the sub-calendar can be published, FALSE otherwise.
PublishCalendarIncludeAttachments	R/W	Boolean. TRUE if attachments to the calendar items should be published to the web; else FALSE.
PublishCalendarIncludePrivateItems	R/W	Boolean. TRUE if items marked private should be published to the web; else False.
PublishCalendarTimePeriodEndDays	R/W	CalendarFolderPublishTimePeriodConstants. If the PublishCalendarTime Period is set to egwPublishLimitedCalendar, the number of days past today that will be published to the web; otherwise this value is ignored.
PublishCalendarTimePeriodStartDays	R/W	CalendarFolderPublishTimePeriodConstants. If the PublishCalendarTime Period is set to egwPublishLimitedCalendar, the number of days previous to today that will be published to the web; otherwise this value is ignored.
PublishEntireCalendar	R/W	Boolean. Set to TRUE if all Calendar contents can be published to the web; else set to FALSE.
Query	R/O	Query. The query associated with a query folder. Applies only to folders whose ObjType = egwQuery.
RSSCameFromIE7	R/W	Boolean. TRUE if RSS feed came from Internet Explorer 7.
RSSClearAllOnRefresh	R/W	Boolean. TRUE if all folder contents are removed, then refreshed from the RSS location.
RSSDoNotCheckImages	R/W	Boolean. TRUE if no warnings about images in items should be given to user.
RSSDownloadPage	R/W	Boolean. TRUE if RSS HTML page shoudl be downloaded.
RSSIsAggregator	R/W	Boolean. TRUE if folder is an RSS aggregator.
RSSIsRoot	R/W	Boolean. TRUE if folder is RSS root.
RSSLocation	R/W	String. The location this folder uses to read an RSS feed.
RSSUpdateFrequency	R/W	FolderUpdateFrequencyConstants. An enumerated value pointing out how often to check the SubscribeCalendarLocation for new items.
Shared	R/O	Enum (SharedFolderConstants). The folder's shared status.
ShowInMainCalendar	R/W	Boolean. If this is not a sub-calendar folder, an error will be returned. If this is a sub-calendar folder, TRUE if the calendar contents will be combined with the main calendar folder; False otherwise.
SubscribeCalendarFrequency	R/W	FolderUpdateFrequencyConstants. An enumerated valuse poinintg out how often to check the SubscribeCalendarLocation for new items.
SubscribeCalendarLocation	R/W	String. The location this folder uses to read an RSS feed.
System	R/O	Boolean. TRUE for Cabinet, Calendar, Mailbox, Root, and Work In Progress folders.

Methods

`Delete()`

Removes this folder from its parent object, deletes this folder and its contained messages, and recursively deletes this folder's subfolders and their contained messages. The deleted messages are added to the Trash. Any folders and messages not owned by this folder's account are not deleted; only this account's references to them are deleted.

`MessageList FindMessages(VARIANT Condition)`

Returns all messages in a folder that pass the filter in Condition. Condition can either be a string or a Filter object. If Condition is a string, it represents a filter expression (see Chapter 5, "Filter Expressions," on page 195). If Condition is a Filter object, it represents a saved filter.

FindMessages is similar to the Find method in the Messages collection, except the Find method in the Messages collection instantiates all messages before applying the filter. To save the overhead of instantiating a messages collection, use this method when you do not already have a messages collection. Once you have a Messages collection, you can call its Find method without losing efficiency.

If the folder is a query folder, FindMessages returns an empty MessageList.

`QuickMessages GetQuickMessagesCollection(Date StartDate, QuickMessagesCreationConstants eHowBuildList)`

Returns a QuickMessages (page 147) collection object. For a list of properties available, see QuickMessage (page 145).

The eHowBuildList parameter controls which message items are included in the collection. The following values are possible for eHowBuildList:

egwNewMessages	Messages items in this account that were created or received only since the StartDate parameter are included
egwModifiedMessages	Message items in this account that have been changed only since the StartDate parameter are included
egwAllMessages	Every message item in this account is added to the QuickMessages collection, and the StartDate parameter is ignored.

The QuickMessages collection should return message items quicker than other ways (such as the Messages collection, using the Messages Find method to retrieve a message list). It is very fast when returning new messages, but retrieving modified messages is somewhat slower. The slowest is to retrieve all messages from an account.

GetQuickMessagesCollection should be called only on a main account. Calling this method from a Proxy account results in an error. It also returns an error if called from the Contacts, Query, or Trash folder.

Messages in a folder that have been shared do not appear in the resulting QuickMessages collection. These folders are not part of the user's database and are outside of the account.

Account (page 26) contains a similar method. However, the Folder object method uses messages that are in this specific folder only, whicle the Account object method looks through every message in the user's database.

NOTE: This method does not return hidden messages. Call the GetQuickMessagesCollectionExt method to return hidden messages.

QuickMessages GetQuickMessagesCollectionExt (Date StartDate, QuickMessagesCreationConstants, eHowBuildList, BOOL bIncludeHidden)

Returns a QuickMessages collection object. The bIncludeHidden parameter dictates whether Hidden messages are included with the returned collection.

The eHowBuildList parameter controls which message items are included in the collection. The following values are possible for eHowBuildList:

egwNewMessages	Messages items in this account that were created or received only since the StartDate parameter are included
egwModifiedMessages	Message items in this account that have been changed only since the StartDate parameter are included
egwAllMessages	Every message item in this account is added to the QuickMessages collection, and the StartDate parameter is ignored.

This method should return message items more quickly than other ways (such as using the AllMessages collection and querying objects with their Message Lists). Retrieving Modified Messages is slower than returning New Messages. Retrieving All Messages from an Account is the slowest.

Move(Folder DestFolder)

Moves this folder to the folder specified by DestFolder. In other words, makes this folder a sub-folder of DestFolder. The Mailbox, Calendar, and Query folders are invalid DestFolder values.

Refresh()

Forces this Folder object and its associated FolderRights and Query objects to reread property values from the message database.

Remarks

The PeekModeFlagConstants can be used by a Trusted Application to set teh appropriate statuses when a Message is opened. The following values are part of a bitmask and can be combined to form the set of flags that are automatically marked on a Message when in Peek Mode:

0 egwNoPeekModeFlags

1 egwPeekModeOpenedFlag

2 egwPeekModeReadFlag

4 egwPeekModeDownloadedFlag

When an Appointment is accepted, it is automatically moved to the Calendar folder. Also, personal Appointments and Tasks go into the Calendar folder by default.

To share a folder, create a FolderRights object for the user (see the Add method in the FolderRightsCollection). To unshare a folder, delete the user's FolderRights objects (see the Delete method in the FolderRights object). Only personal folders (ObjType = egwPersonalFolder) can be shared.

For an unshared folder (Shared = egwNotShared), the FolderRights collection is empty.

For a shared folder owned by the user (Shared = egwSharedOutgoing), the FolderRights collection contains one or more read/write instances of FolderRights objects, each of which grants access rights to a specific Address. For a shared folder owned by someone else (Shared = egwSharedIncoming), the FolderRights collection contains the FolderRights for all users.

When a Folder is refreshed, it recursively refreshes the Folders collection returned by its AllFolders property, its FolderRights collection, Folders collection, Messages collection, the Address object returned by its Owner property, and its Query object.

The list of System folders includes the Contacts, Sent Items, and Checklist folders.

The Contacts folder is a shortcut to an Address Book. In the GroupWise client, a user is free to add, delete, or modify any Address Book Entry. In the Object API, the Contacts folder returns an exception if it is asked for its Messages collection, or if a program attempts to move Messages into the folder.

In the place of the Messages Collection, use the ContactsAddressBook property to retrieve the Address Book that the Contacts folder is currently using. Then retrieve the AddressBookEntries collection from the AddressBook as the items in this folder.

By default, the Contacts folder works with the Frequent Contacts address book. To change the default setting, retrieve any GroupWise Personal AddressBook object and set the Contacts folder ContactsAddressBook property with this Address Book.

The Sent Items and Task List folders are not created by default. However, a GroupWise user can still create these folders.

The Sent Items system folder replaces the Sent Items query folder. Retrieving the Sent Items system folder Messages collection returns items Sent by this user to others that are also contained in the Mailbox, Sent Items, or Calendar folder.

NOTE: Unlike the Sent Items query folder, the Sent Items system folder does not do a Query to obtain the list of contents. This step should improve the response time when you are retrieving the Messages collection from this folder.

The Checklist folder contains a special group of Message items. These items have been moved into the Checklist folder, or these items remain in their folder but have been marked with the Show On Checklist flag. Please see the Appointment, DocumentReference, Mail, Note, Phone, Task, and SharedNotification objects for further Checklist features.

GroupWise 8.0 has added the ability to publish or subscribe calendars through a Web service. These properties allow a third party to modify the settings on an individual sub-calendar.

Calendar publishing is accomplished using a new Web Service. Any sub-calendar that is allowed will be published to the given location. Please see the calendar publishing web service documentation for further details.

On the other hand, subscription calendars settings are used only by GroupWise clients. Items are read from the given location, then displayed by GroupWise clients. No items are created in the GroupWise Database. As such, the Object API will return the subscription calandar, but the messages collection of that calendar will be empty.

NOTE: A GroupWise Administrator must enable calendar publishing and subscriptions before these properties can be used. If the administrator disables calendar publishing or subscriptions, all methods related to that feature will return an “Access Denied” error.

FolderRights

Provides access rights to a folder.

Properties

The following table lists properties along with their access and descriptions.

Property	Access	Description
Address	R/O	Address. The address of the user to whom access rights are being granted.
AllowAdd	R/W	Boolean. TRUE if the user can add items to the folder.
AllowDelete	R/W	Boolean. TRUE if the user can remove items from the folder.
AllowModify	R/W	Boolean. TRUE if the user can modify items in the folder.
Application	R/O	Application. The Application object.
BitMask	R/W	Long. A bit mask that contains one bit for each Boolean property. See FolderRightsConstants for the bit definitions. This property is synchronized with the Boolean properties.
Parent	R/O	FolderRightsCollection. The FolderRightsCollection that owns this object.

Methods

`Delete()`

Removes this FolderRights object from its owning FolderRightsCollection, revokes the user's access rights to the shared folder, and deletes the FolderRights object itself. When the last FolderRights object is deleted from a FolderRightsCollection, the owning folder ceases to be a shared folder. To unshare a folder, you must call the Commit method of the FolderRightsCollection after calling this Delete method.

Remarks

FolderRights properties are writable only by the folder owner.

A FolderRights object is refreshed when its parent object is refreshed. When a FolderRights object is refreshed, it recursively refreshes its Address object.

System folders cannot be shared with others users. The list of System folders includes the Mailbox, Calendar, Contacts, Sent Items, and Contacts folders.

FolderRightsCollection

A collection of FolderRights objects.

Properties

The following table lists properties along with their access and descriptions.

Property	Access	Description
Application	R/O	Application. The Application object.
Count	R/O	Long. The number of objects in this collection.
_NewEnum	R/O	Enumeration object. Implements IEnumVARIANT. For Windows only.
Parent	R/O	Folder. The Folder object that owns this collection.

Methods

`Add(VARIANT Address, Long Rights)`

If Address is an Address object, this method creates a new FolderRights object for that address. If Address is an Addresses collection, this method creates a new FolderRights object for each address in the collection. Rights specifies the rights to be given. The value for Rights is derived from combining FolderRightsConstants with the bitwise inclusive OR operator.

- Because there is seldom need to access a new FolderRights object immediately after it is created, this Add method does not return the newly created object.
- Because only personal folders can be shared, this method throws an exception if the owning folder's ObjType property is not egwPersonal.

Passing an Address object that refers to a group will expand the group and add each contained user individually.

To share a folder, you must call the Commit method after calling this Add method.

`Commit(String Subject, String BodyText, [FolderRightsNotifyConstants TypeCommit])`

Commits the changes made to the collection.

- Unlike other collections in the Object API, the FolderRightsCollection does not save changes unless the Commit method is called. Calling this method will cause a notification message to be sent to the affected users. For example, if you add users to this collection, calling Commit will send the sharedfolder message to those users.
- TypeCommit indicates what changes are being committed. If TypeCommit is omitted, all shared users are notified.

`FolderRights Item(VARIANT Index)`

DEFAULT. Returns the FolderRights object specified by Index. Index may be a Long, a string representing a Long, or an Address object. If Index is a Long, or a string representing a Long, returns the FolderRights object located at the given Index in the collection. Valid indexes are 1 through Count, inclusive. Throws an exception if the Index is outside of this range. If Index is an Address object, returns the FolderRights object whose Address is equal to Index.

NOTE: The FolderRightsCollection essentially has a distribution list (or a list of AddressBookEntries (page 39)) of users that have shared rights to the folder in question. Because this method expects an AddressBookEntry (page 41) that is in this distribution list, you can pass in an AddressBookEntry of a user that has shared rights for the folder only.

Remarks

A FolderRights collection is refreshed when its parent object is refreshed. When a FolderRights collection is refreshed, it recursively refreshes its contained FolderRights objects.

Adding the first FolderRights object to a FolderRights collection changes the owning folder to a shared folder (the folder's Shared property changes to egwSharedOutgoing). Deleting the last FolderRights object changes the folder back to unshared (the folder's Shared property changes to egwNotShared). All of these changes occur after the Commit method is called.

Folders

Contains acollection of Folder objects.

Properties

The following table lists properties along with their access and descriptions.

Property	Access	Description
Application	R/O	Application. The Application object.
Count	R/O	Long. The number of objects in this collection.
_NewEnum	R/O	Enumeration object. Implements IEnumVARIANT. For Windows only.
Parent	R/O	Object. The Folder or Message object that owns this collection.

Methods

`Folder Add(SharedNotification SharedMessage, [String Name])`

Creates a shared folder based on the information stored in the SharedNotification object SharedMessage. The folder can optionally be named by passing in the Name parameter. SharedNotification messages are generated by the Commit( ) method of the FolderRightsCollection and have a class name of GW.MESSAGE.MAIL.NGW.SHARED.FOLDER.NOTIFY.
Calling this method causes the original shared folder notification message to be deleted. References to the shared folder notification message should be immediately released after calling this Add method. Query and system folders cannot be parents for other folders. Invoking the Add method on these folders will throw an exception.

The Add method does not allow child folders to be created under the Mailbox, Calendar, Contacts, Sent Items, or Checklist folders. Also, the Add method does not allow child folders to be created under any Query folder.

This method adds a folder only at the current folder level and won't accept path arguments that contain the backslash (\) character. If you want to add a folder to a subfolder level (such as beneath the Cabinet folder), you must first get the Folders collection beneath the top level folder (for example, the Cabinet folder) and then call the Add method.

`Folder Add(String Name)`

Creates a new personal folder. To create a new query folder, see the Query object. Query and system folders cannot be parents for other folders. Invoking the Add method on these folders will throw an exception.

`Folder Item(VARIANT Index)`

DEFAULT. Returns the Folder object specified by Index. Index may be a Long or a string. If Index is a Long, returns the Folder object located at the given Index in the collection. Valid indexes are 1 through Count, inclusive. Throws an exception if the Index is outside of this range. If Index is a string, returns the Folder object whose FolderID is equal to Index.

`Folder ItemByName(String Name)`

Returns the Folder object specified by Name.

AddSubCalendar(String sNewFolderName, Boolean bShowInMainCalendar)

Creates a new sub-calendar folder. This function can only be called from the folders collection of the main calendar folder; all other folders collection objects will return an error. The new folder name may not contain the characters ":","/", or "\". The new folder name cannot match any current sub-calendar name. If all tests are passed, the new sub-calendar will be created with the given name. In addition, the new sub-calendar ShowInMainCalendar flag will be set.

AddSubscribeCalendar(String sNewFolderName, Boolean bShowInMainCalendar, String sSubscriberLocation, FolderUpdateFrequencyConstants eSubscriberFrequency)

Creates a new subscription calendar folder. This function can only be called from the Folders collection of the main calendar folder; all other folders collection objects will return an error. The new folder name may not contain the characters ":", "/", or "\". The new folder name cannot match any current subscriptions or sub-calendar name. If all tests are passed, the new subscription calendar will be created with the given name. The new subscription folder will be created with the ShowInMainCalendar flag set, the given location as the URL or WebCal to retrieve calendar information from, and with the update frequency set to the eSubscribeFrequency.

NOTE: This folder will be treated the same as the GroupWise Windows client. No actual calendar items will be placed in the GroupWise database. This will simply read the calendar items from the subscription location, then display them in the GroupWise client. If using the Object API, the messages collection obtained from this folder will not contain any items.

Remarks

The Add method template is as follows:

Add(VARIANT P1, [VARIANT P2]).

The Add method checks the P1 type at runtime.

- If P1 is a SharedNotification object, the first form of Add is assumed; P2, if present, must be a string.
- If P1 is a String, the second form of Add is assumed; P2 if present, is ignored.

The Add method does not allow child folders to be created under the Mailbox, Calendar, Contacts, Sent Items, or Checklist system folders, or under any Query folder.

A Folders collection is refreshed when its parent object is refreshed. When a Folders collection is refreshed, it updates the Folder objects in the collection, but it does not recursively refresh the Folder objects themselves.

Folders2

Contains a collection of Folder objects.

Properties

The following table lists properties along with their access and descriptions.

Property	Access	Description
Application	R/O	Application. The Application object.
Count	R/O	Long. The number of objects in this collection.
_NewEnum	R/O	Enumeration object. Implements IEnumVARIANT. For Windows only.
Parent	R/O	Object. The Folder or Message object that owns this collection.

Methods

`Folder Add(SharedNotification SharedMessage, [String Name])`

Creates a shared folder based on the information stored in the SharedNotification object SharedMessage. The folder can optionally be named by passing in the Name parameter. SharedNotification messages are generated by the Commit( ) method of the FolderRightsCollection and have a class name of GW.MESSAGE.MAIL.NGW.SHARED.FOLDER.NOTIFY.
Calling this method causes the original shared folder notification message to be deleted. References to the shared folder notification message should be immediately released after calling this Add method. Query and system folders cannot be parents for other folders. Invoking the Add method on these folders will throw an exception.

The Add method does not allow child folders to be created under the Mailbox, Calendar, Contacts, Sent Items, or Checklist folders. Also, the Add method does not allow child folders to be created under any Query folder.

This method adds a folder only at the current folder level and won't accept path arguments that contain the backslash (\) character. If you want to add a folder to a subfolder level (such as beneath the Cabinet folder), you must first get the Folders collection beneath the top level folder (for example, the Cabinet folder) and then call the Add method.

`Folder Add(String Name)`

Creates a new personal folder. To create a new query folder, see the Query object. Query and system folders cannot be parents for other folders. Invoking the Add method on these folders will throw an exception.

`Folder Item(VARIANT Index)`

DEFAULT. Returns the Folder object specified by Index. Index may be a Long or a string. If Index is a Long, returns the Folder object located at the given Index in the collection. Valid indexes are 1 through Count, inclusive. Throws an exception if the Index is outside of this range. If Index is a string, returns the Folder object whose FolderID is equal to Index.

`Folder ItemByName(String Name)`

Returns the Folder object specified by Name.

Refresh()

Forces the object in a Folders collection to reread all subfolders from the database that belong to this collection. Note that new folders might have been added to the collection and old folders might have been deleted.

Remarks

The Add method template is as follows:

```
Add(VARIANT P1, [VARIANT P2]).
```

The Add method checks the P1 type at runtime.

- If P1 is a SharedNotification object, the first form of Add is assumed; P2, if present, must be a string.
- If P1 is a String, the second form of Add is assumed; P2 if present, is ignored.

The Add method does not allow child folders to be created under the Mailbox, Calendar, Contacts, Sent Items, or Checklist system folders, or under any Query folder.

A Folders collection is refreshed when its parent object is refreshed. When a Folders collection is refreshed, it updates the Folder objects in the collection, but it does not recursively refresh the Folder objects themselves.

FormattedText

Represents text.

Properties

The following table lists properties along with their access and descriptions.

Property	Access	Description
Application	R/O	Application. The Application object.
Parent	R/O	Message. The Message object that owns this object.
PlainText	R/W	DEFAULT. String. Plain text without embedded formatting codes. The message body field has a limit of 32,000 characters.
RTF	R/W	String. The text with embedded formatting codes - Rich Text Format (RTF). The message body field has a limit of 32,000 characters.
Unicode	R/O	String. The Unicode version of the text.

Remarks

A FormattedText object is refreshed when its parent object is refreshed.

GroupMember

Represents a member of a group. A subtype of AddressBookEntry.

Properties

The following table lists properties along with their access and descriptions.

Property	Access	Description
TargetType	R/O	Enum (AddressTargetTypeConstants). The target type (recipient type) of the group member.
Parent	R/O	GroupMembers collection. The GroupMembers collection that own this object.

Methods

`Delete()`

Deletes this GroupMember object from the GroupMembers collection.

`Refresh()`

Forces this GroupMember object to reread property values from the message database.

GroupMembers

A collection of GroupMember objects.

Properties

The following table lists properties along with their access and descriptions.

Property	Access	Description
Application	R/O	Application. The Application object.
Count	R/O	Long. The number of objects in this collection.
_NewEnum	R/O	Enumeration object. Implements IEnumVARIANT. For Windows only.
Parent	R/O	AddressBookEntry. The AddressBookEntry object that owns this collection.

Methods

`Add(VARIANT AddressBookEntry, [AddressTargetTypeConstants TargetType])`

Adds an existing address book entry to this collection and to the associated group. You cannot add an Address object to this collection. Group members are required to be AddressBookEntry objects and these AddressBookEntry objects must have a valid email address. The TargetType parameter specifies the target type (recipient type) to assign the address or addresses in the group. If TargetType is omitted, egwTo is assumed. See AddressTargetTypeConstants.

`GroupMember Item(VARIANT Index)`

DEFAULT. Returns the GroupMember object located at the given Index in the collection. Index may be a Long or a string representing a Long. Valid indexes are 1 through Count, inclusive. Throws an exception if the Index is outside of this range.

Remarks

A GroupMembers collection is refreshed when its parent object is refreshed. When the GroupMembers collection is refreshed, it recursively refreshes its contained Address objects.

GWTimeZone

A collection of GWTimeZone objects.

Properties

The following table lists properties along with their access and descriptions.

Property	Access	Description
Application	R/O	Application. The Application object.
DSTName	R/W	String. The name of this timezone when daylight savings time is in effect.
DSTOffsetFromStandard	R/W	Long. The number of minutes that daylight savings timemoves from the StandardOffsetFromGMT.
DSTYear	R/W	Integer. The specific year that the daylight savings time will apply. If zero, the month and day for the start and end of daylight savings time will be computed.
EndDSTDay	R/W	Integer. The day of the month that daylight savings timewill end. For instance, the 4th of the month.
EndDSTDayOfTheWeek	R/W	TimeZoneDaysOfWeekConstants. The day of the week during a month that daylight savings time will end. For example, the first Saturday of the month.
EndDSTMonth	R/W	TimeZoneMonthConstents. The month when daylight savings time ends. For instance, November.
EnsDSTOccurance	R/W	TimeZoneOccuranceConstants. The occurance of the day of the week during a month that daylight savings time will end. For example, the first Saturday of the month.
Parent	R/O	Account. The account object that ownes this TimeZone.
StandardName	R/W	String. The name of this time zpne when daylight savings time is not in effect.
StandardOffsetFromGMT	R/W	Long. The number of minutes tha this timezone differs from Greenwich Mean Time. For example, the Mountain Time Zone will return -420. This equates to 7 hours less than GMT.
StartDSTDay	R/W	Interger. The day of the month that daylight savings time will start. For example the 11th of the month.
StartDSTDayOfTheWeek	R/W	TimeZoneDaysOfTheWeekConstants. The day of the week that daylight savings time will start. For example, Sunday.
StartDSTMonth	R/W	TimeZoneMonthConstants. The month when daylight savings time will start. For example, March.
StartDSTOccurance	R/W	TimeZoneOccuranceConstant. The occurance of the day of the week during a month that daylight savings time starts. For example, the second Sunday of the month.

Methods

`Commit()`

Comits all chamges to the time zoneto the GroupWise account. On success, all new appointments, tasks, or notes created will contain the TimeZone information given.

Remarks

All dates in the GroupWise System are stored in Greenwich Mean Time (GMT). When writing dates to the database, the system converts the given date to GMT using the Time zone of the machine. When retrieving dates, the Time zone is applied in reverse.

Starting with the 6.5 version of the GroupWise Windows Client, all Group Appointments, Notes, and Tasks had the Time zone used, when the item was first created, also stored in the database.

Starting with release 8.0, the GroupWise Windows Client will store all Appointment, Notes, and Tasks with the Time zone information. Furthermore, the GroupWise Object API has been enhanced to also store the Time zone information.

For most application, the fact that a time zone is used is not important. GroupWise recommends that Third Parties not worry about this data.

If needed, the GroupWise 8.0 release of the Object API will allow a Third Party to read the time zone information used when creating these calendar items. Calling the new GWTimeZone property will return an object with all needed data.

Further, if a Third party is interfacing with a mobile device or system in another time zone, they may also set the properties of the GWTimeZone object. When the Third Party program calls the Commit method of the GWTimeZone object, all further Appointments, Notes, and Tasks created for this account will have the given Time Zone information stored in the GroupWise database.

Note: The GroupWise Object API still converts dates given using the time zone of the computer. The new GWTimeZone information is stored in the database along with the calendar item.

For more information on time zones and their data, please refer to the GetTimeZoneInformation call and the associated TIME_ZONE_INFORMATION structure in Microsoft Visual C.

IMAddress

Provides an instant messaging address.

Properties

Property	Access	Description
Application	R/O	Application. The Application object.
InstantMessagingAddress	R/O	String. Returns the Instant Messaging Address text of this object.
InstantMessagingAddressType	R/O	IMAddressTypeConstants. Returns the Instant Messaging Address type for this object.
Parent	R/O	AddressBookEntry. Returns the Address Book entry that owns this object.
DefaultAddress	R/O	Boolean. Tthe "default" address for this collection. (Making an IM Address the default means the GroupWise system uses this IM address whenever the parent Address Book entry is asked for the Instant Messaging Address to use.)

Methods

`Delete()`

Removes this IM Address from the parent Address Book Entry.

`MakeDefaultAddress()`

Sets this address as the Default Instant Messaging Address. (There must be a default address for each collection. Therefore, setting an IM Address as the default was coded as a method, not a Property that can have FALSE passed in.)

IMAddresses

Provides an instant messaging address.

Properties

Property	Access	Description
Application	R/O	Application. The Application object.
Count	R/O	Long. The number of items in this collection.
_NewEnum	R/O	Enumeration object. Implements IEnumVARIANT. For Windows only.
Parent	R/O	AddressBookEntry pointer. The Dispatch pointer at the Address Book Entry object that owns this Collection.

Methods

`IMAddress Add(String NewIMAddress, IMAddressTypeConstant NewIMType)`

Adds a new Instant Messaging Address to the collection. The new address is given by the String NewIMAddress passed in. Returns an IMAddress object. Throws an exception if the given IM Address is already in the collection.

`IMAddress Item(Long Index)`

Default. Returns the IMAddress object located at the given Index in the collection. Valid indexes are 1 through Count, inclusive. Throws an exception if the index is outside of this range.

Locations

A collection of Accounts, DocumentLibraries, and Folders for a query to search.

Properties

The following table lists properties along with their access and descriptions.

Property	Access	Description
Application	R/O	Application. The Application object.
Count	R/O	Long. The number of objects in this collection.
_NewEnum	R/O	Enumeration object. Implements IEnumVARIANT. For Windows only.
Parent	R/O	Query. The Query object that owns this collection.

Methods

`Add(Object Location)`

Adds Location to the collection. Location must be an Account object (searches account folders), DocumentLibrary object (searches library document versions), or Folder object (searches folder). If Location is a Folder object, it cannot be a Calendar or Query folder. To search all folders without listing them, search the root account.

`Object Item(Long Index)`

DEFAULT. Returns the Account, DocumentLibrary, or Folder object located at the given Index in the collection. Valid indexes are 1 through Count inclusive. Throws an exception if the Index is outside of this range.

`Remove(Long Index)`

Removes the location object located at the given Index, from the collection. This method will not delete or destroy the location object. It merely removes the object from the collection.

Remarks

A Locations collection is refreshed when its parent object is refreshed. When a Locations collection is refreshed, it updates the objects in the collection, but does not recursively refresh the objects themselves.

LookupTableEntries

A collection of LookupTableEntry objects.

Properties

The following table lists properties along with their access and descriptions.

Property	Access	Description
Application	R/O	Application. The Application object.
Count	R/O	Long. The number of objects in this collection.
_NewEnum	R/O	Enumeration object. Implements IEnumVARIANT. For Windows only.
Parent	R/O	FieldDefinition. The FieldDefinition object that owns this collection.

Methods

`LookupTableEntry Item(Long Index)`

DEFAULT. Returns the LookupTableEntry object located at the given Index in the collection. Valid indexes are 1 through Count inclusive. Throws an exception if the Index is outside of this range.

LookupTableEntry

An entry in a lookup table. Lookup table entries are used to provide restricted values for field.

Properties

The following table lists properties along with their access and descriptions.

Property	Access	Description
Application	R/O	Application. The Application object.
Archived	R/W	Boolean. TRUE if this message has been archived.
Parent	R/O	LookupTableEntries. The LookupTableEntries collection that owns this object.
Value	R/O	VARIANT. The value of this entry. Will be a number or string.

Mail

Provides information and actions for routed messages. A subtype of Message.

Properties

The following table lists properties along with their access and descriptions.

Property	Access	Description
Archived	R/W	Boolean. TRUE if this message has been archived.
Categories	R/O	Categories Collection. The collection of Category objects on this Mail, Appointment, Document Reference, Note, Phone Message, Task, or AddressBookEntry.
ChecklistCompleted	R/W	Boolean. TRUE if the item is part of the Checklist folder Messages collection or a Task item AND the item has been marked Completed. FALSE otherwise.
ChecklistCompletedDate	R/W	Date. The date this item was marked completed.
ChecklistDueDate	R/W	Date. The date this item should be Completed. Returns an error if the Message item is not a Task or is not part of the Checklist folder Messages collection.
ChecklistPercentComplete	R/W	Integer. Between 0 and 100 percent, how complete the checklist item is. If this value is set to a number outside the 0 to 100 range, an error is returned. If the value is set to 100, the item will also be marked complete.
ChecklistShowOnCalendarDate	R/W	Date. The date this item should appear on the calendar task list.
ClientMessageID	R/O	The ClientMessageId method will return the string that the GroupWise Client puts in the properties page of an item.
CommonMessageID	R/O	String. The ID of the message, regardless of its location. For example, an email will have the same CommonMessageID for both Sent and Received.
Completed	R/W	Boolean. TRUE if the current user is done with this message and it's ready to be sent to the next user in the recipient list.
DelayedDeliveryDate	R/W	Date. When the Message.Send() method is called, the message is not delivered until the specified Date. This property can be used only on a Draft Message.
Delegated	R/O	Boolean. TRUE if this message was delegated to you.
DeliveredDate	R/O	Date. The date and time this message item was delivered to the account.
Forwarded	R/O	Boolean. TRUE if this item has been forwarded to another user. Otherwise, FALSE.
LastModifiedRetentionDate	R/O	Normally an item is retained after a set amount of time. It is possible to add personal notes or personal attachments on items. The LastModifiedRetentionDate is used for retention software to know that significant data has changed on the item and that the retention software needs to capture the item again.

Property	Access	Description
MessageProperty	R/O	String. Based on the MessagePropertiesConstant passed in, a different string is returned.
NotifyWhenCompleted	R/W	Enum (NotifyMessageConstants). Specifies the type of notification to send when this message has been marked completed.
PersonalSubject	R/W	String. A text string to be shown by the GroupWise Windows client as the Subject text. This text appears only for the current user. The normal subject or their own Personal Subject text appears for other users.
Replied	R/O	Boolean. TRUE if this item has been replied to. Otherwise, FALSE.
Sharer	R/O	Address. The user who added this message to a shared folder. If this message was not added to a shared folder, this will be the same as the Sender property.
ShowOnChecklist	R/W	Boolean. TRUE if the item is part of the Checklist folder Messages collection. FALSE otherwise.
Size	R/O	Returns the size of the attachments and message body. If there are no attachments, only the size of the message body is returned.

Methods

`GetMIME ()`

Some developers will store the GroupWise data as MIME. The GetMIME method will return all GroupWise items whether or not it has a MIME/RFC822 attachment. This method is very CPU intensive and should be avoided unless absolutely needed.

`LocalDelete()`

Deletes this message only from the currently connected post office database. No synchronization will take place between the master database and a remote database.

`Mail Delegate()`

Delegate this message to another user. Returns the OutBox mail message.

`RemoveChecklistDueDate()`

Removes the Checklist Due Date from the Message item. Returns an error if the item is not on the Checklist.

Message

Base class containing methods and properties common to items that may be sent to a user.

Properties

The following table lists properties along with their access and descriptions.

Property	Access	Description
Application	R/O	Application. The Application object.
Attachments	R/O	Attachments collection. The collection of attachments to this message.
BodyText	R/O	FormattedText. The body text of this message. The message body field has a limit of 32,000 characters.
BoxType	R/O	Enum (MessageBoxTypeConstants). The box type of this message.
ClassName	R/O	String. GroupWise names its messages with the following default class names: GW.MESSAGE.APPOINTMENT GW.MESSAGE.DOCUMENTREFERENCE GW.MESSAGE.MAIL GW.MESSAGE.NOTE GW.MESSAGE.PHONE GW.MESSAGE.TASK Custom 3rd-Party Object (C3PO) servers can further subtype these names by appending further classifications. For example, A C3PO might create GW.MESSAGE.TASK.PROJECTX.
ConvertAttachments	R/W	Boolean. TRUE if we can convert the attachments.
CreationDate	R/O	Date. The date and time this message was created.
Deleted	R/O	Boolean. TRUE if this message has been deleted (is in the Trash). A message can be simultaneously in the Trash and in other folders.
DownloadStatus	R/O	DownloadStatus. Provides information about the message parts that have been downloaded. When the account containing the message is not a remote account, this property is still available and always indicates that everything has been downloaded.
EnclosingFolders	R/O	Folders collection. Returns the folder this item is contained in.

Property	Access	Description
ExpandedRecipients	R/O	Recipients collection. This property returns an expanded set of recipients for the message. The expanded set is the set of recipients associated with the message once all groups have been expanded. For draft messages, expansion of the groups occurs when this property is accessed. For InBox or OutBox messages, the groups were expanded when the message was originally sent. The returned collection may not contain every recipient of the message. If a message is sent to a system group on a foreign domain, GroupWise is unable to track the list of users. The foreign group will not be expanded. This collection is informational only. Adding a recipient to the collection does not cause the message to be sent to that recipient. Use the Recipients property for addressing a message.
Fields	R/O	Fields collection. The collection of fields for this message.
FromText	R/W	String. The text provided in the From field for the message. The content defaults to the full name of the sender. If the From text is modified, the user's full name (in parentheses) is always appended to the text.
Hidden	R/W	Boolean. TRUE if this message is hidden. If this message is hidden, it will not show up in any User Interface (UI).
MasterRevisionNumber	R/O	Long. The revision number of the message when the message was downloaded in remote.
MessageID	R/O	String. A unique, persistent (valid across sessions) ID for this message.
ModifiedDate	R/O	Date. The date and time this message was last modified.
NotifyWhenDeleted	R/W	Enum (NotifyMessageConstants). Specifies the type of notification to send when this message has been deleted.
NotifyWhenOpened	R/W	Enum (NotifyMessageConstants). Specifies the type of notification to send when this message has been opened.
Opened	R/W	Boolean. TRUE if this message has been opened. Attempting to set this property to FALSE would be a violation of GroupWise security, so that is not allowed. Setting this property to TRUE does not affect the value of the Read property.
Parent	R/O	Account. A reference to the Account object. A message can exist in more than one folder, so there is no natural parent.
Priority	R/W	Enum (MessagePriorityConstants). The priority of this message
Private	R/W	Boolean. TRUE if this message is private (i.e. a proxy user cannot see this message).
Read	R/W	Boolean. TRUE if this message has been read.
Recipients	R/O	Recipients collection. The collection of users and groups to whom this message is sent.
ReplyChildren	R/O	MessageList. The list of messages that are next in line in a reply thread.
ReplyDaysRequested	R/W	Long. The number of days within which a reply is requested.
ReplyParent	R/O	Message. The message before this message in the reply thread.
ReplyRequested	R/W	Enum (ReplyRequestedConstants). Specifies if and when a reply is requested.

Property	Access	Description
ReplyRoot	R/O	Message. The first message in a reply thread.
RevisionNumber	R/O	Long. The number of times this message has been revised.
Routed	R/W	Boolean. TRUE if this message has been routed.
RoutingEndOfLine	R/O	Boolean. TRUE if this message has reached the last user in the routing list.
Sender	R/O	Address. The user who sent this message.
Subject	R/O	FormattedText. The subject of this message.
ViewName	R/W	String. The name of the view associated with this message.

Methods

`Annotate(Note Note)`

Adds an existing personal Note object as an attachment to this message. Works even if this message has been sent, because the Note is personal. See also the Add method in the Attachment object.

- Annotating an item requires create and modify rights to the folder that contains the item.
- Annotating encapsulated items is not allowed because no distinct containing folders exist for such an item.
- Annotations can be edited only by the user who originally created the annotation (subject also to folder rights). For example, an annotation created on an item in a shared folder can only be edited by the user who created the annotation.

`Message Clone()`

Makes an independent copy of this message and returns it as a new draft message. The new message is contained in the same folders as the original.

`Delete()`

Removes a message from folders and puts it in the Trash. There will be one Trash entry for each folder that contained the message. Deleting a message places the message into the owner's trash. This can be somewhat confusing when deleting messages from a shared folder. The message is placed in the owner's trash and not necessarily into the trash of the account that made the delete call.

`Mail Forward()`

Forwards this message. Returns a new draft Mail object with this Message object included as an attachment. The new message is contained in the same folders as the original.

`Refresh()`

Forces this message and associated objects and collections to reread property values from the message database. The actual reading of a specific property may be postponed until the next time the property is accessed. This "lazy evaluation" is an optimization that avoids unnecessary reading of unaccessed properties. If the message is an attachment, it is refreshed when its associated Attachment object is refreshed.

Message Reply([String ClassName], [Boolean ReplyToAll], [Boolean IncludeSenderMessageText], [Boolean AttachOriginalMessage])

Creates a reply message. Returns a new draft Message object item with the message sender as recipient.

- ClassName specifies the kind of message to create. If ClassName is omitted, GW.MESSAGE.MAIL is assumed.
- If ReplyToAll is set to TRUE, all recipients of this message, except the current user, will be included as recipients in the reply message.
- If IncludeSenderMessageText is set to TRUE, includes this message's BodyText in the reply message's BodyText.
- If AttachOriginalMessage is set to TRUE, attaches this message to the reply message.

Any omitted Boolean parameters are defaulted to FALSE. The reply message is contained in the same folders as the original. The reply message is not automatically linked to the Work In Progress folder (unless the original message was linked there).

Retract()

Requests message retraction from recipient mailboxes. The message's BoxType must be egwOutgoing.

Message Send()

Returns the OutBox message after this message is sent. This method will delete the draft message to be deleted. When recipients fail to resolve, this method will throw an exception. This method does not update the Frequent Contacts address book. This should be done manually by the caller of the method.

Procedures

To move a message between folders, move it from the old folder's Messages collection to the new folder's Messages collection.

To move a message to the Trash, delete it from the folder's Messages Collection.

To undelete a Trash message, add it back to its old Messages collection.

To link a message to folders, add it to the Messages Collections of the desired folders.

To unlink a message from a folder, delete it from the folder's Messages Collection.

To empty a message from the Trash, delete the TrashEntry object.

To empty the Trash, use the Trash object's Empty method.

Remarks

When folder A and folder B contain a message, the message's EnclosingFolders property includes A and B. The Messages collections of folders A and B contain the message. When the message moves from A to the Trash, the message's Deleted property is TRUE, and its EnclosingFolders property contains only folder B. When the message moves from B to the Trash, its Deleted property remains TRUE, and its EnclosingFolders property contains nothing

When a Message object is refreshed, it recursively refreshes its Attachments collection, the FormattedText objects returned by its BodyText and Subject properties, its Fields collection, its Recipients collection, and the Address object returned by its Sender property. It also updates

Message objects returned by its ReplyChildren, ReplyParent, and ReplyRoot properties, but it does not recursively refresh those messages. Specific subtypes of the Message object may recursively refresh some of their own properties.

MessageList

An independent list of messages that does not persist in the message database. This list is a snapshot of a list of messages from the message database.

Properties

The following table lists properties along with their access and descriptions.

Property	Access	Description
Application	R/O	Application. The Application object.
Count	R/O	Long. The number of objects in this collection
_NewEnum	R/O	Enumeration object. Implements IEnumVARIANT. For Windows only.
Parent	R/O	Application. A reference to the Application object. Because this is an independent list, it has no natural parent.

Methods

`Add(VARIANT Message)`

Adds an existing message or list of messages to this collection. Message may be a Message object, a MessageList collection, or a string. If Message is a Message object, the message is added. If Message is a MessageList collection, each Message in that collection is added. If Message is a string, the message whose message ID is equal to the Message is added.

`MessageList Find(VARIANT Condition)`

Returns the MessageList collection containing the messages matching the given Condition. Condition may be a string or a Filter object. If Condition is a string, it represents a filter expression. See Chapter 5, “Filter Expressions,” on page 195. If Condition is a Filter object, it represents a saved Filter.

`Message Item(VARIANT Index)`

DEFAULT. Returns the Message specified by Index. Index may be a Long or a string. If Index is a Long, returns the Message object located at the given Index in the collection. Valid indexes are 1 through Count, inclusive. Throws an exception if the Index is outside of this range. If Index is a string, returns the Message object whose MessageID is equal to Index.

`Remove(VARIANT Index)`

Removes the message specified by Index from this collection. Index may be a Long, a string, or a Message object. If Index is a Long, removes the Message object located at the given Index in the collection. Valid indexes are 1 through Count, inclusive. Throws an exception if the Index is outside of this range. If Index is a string, removes the Message object whose MessageID is equal to Index. If Index is a Message object, removes the given Message object from the collection.

Remarks

A MessageList collection does not need to be refreshed because it is independent of the message database.

Messages

A collection of Message objects.

Properties

The following table lists properties along with their access and descriptions.

Property	Access	Description
Application	R/O	Application. The number of objects in this collection.
Count	R/O	Long. The number of objects in this collection.
_NewEnum	R/O	Enumeration object. Implements IEnumVARIANT. For Windows only.
Parent	R/O	Folder. The Folder object that owns this collection.

Methods

`Message Add([String ClassName], [Enum (egwDraft, egwPersonal) ObjType])`

Creates a new Message and links it to the folder that owns this collection. ClassName is a C3PO class name that specifies the kind of message to create. If ClassName is omitted, a mail message (GW.MESSAGE.MAIL) is assumed.ObjType specifies whether to create a draft or personal message. If ObjType is omitted, egwDraft is assumed.

`Message Add([String ClassName], [Object Referent], [DocumentReferenceTypeConstants Version])`

Creates a new Message and links it to the folder that owns this collection.

- ClassName is a C3PO class name that specifies the kind of message to create. If ClassName is omitted, a document reference (GW.MESSAGE.DOCUMENTREFERENCE) is assumed. .
- Referent is the object the new message refers to. For GroupWise, Referent applies only to messages of class GW.MESSAGE.DOCUMENTREFERENCE and its subclasses where Referent must be either a Document or DocumentVersion object. Referent is required when creating a document reference.
- Version controls how the document reference refers to the document or document version. If Referent is a Document object and Version is omitted, egwCurrent is assumed. If all parameters are omitted, the first version of the Add method is assumed.

`Message Add(Message Message)`

Adds an existing Message object to this collection and links it to the folder that owns this collection. A message can exist in multiple folders.

- If there is a corresponding Trash entry, it is automatically deleted.
- The Message object used as a parameter must be of type VARIANT. If early binding is used to dimension the message object, an error might result.
- If Message and the folder that owns this collection do not belong to the same account, an exception is thrown.

When the Messages collection represents a shared folder, calling this method can cause a copy of the message to be added. That is, the returned message may be a copy of the message being added rather than simply returning the parameter passed in. This is because moving an item to a

shared folder often results in the item being copied to a unique "prime" database. As a result, rather than a single message in two folders, there are two unique messages, each in a unique folder.

```
Message AddExistingMessage (String SenderDisplayName, String SenderEmailAddress,
String SenderEmailAddressType, Date CreationDate, MessageBoxTypeConstants
MessageBoxType, MessageStatusConstants MessageStatus, MessagePriorityConstants
MessagePriority, MessageSecurityConstants MessageSecurity, [VARIANT DraftMessage],
[VARIANT LastModificationDate])
```

Adds a Message to the database, but does not send the message. The GroupWise system treats these like InBox or OutBox items but does not try to send them. This method is useful for synchronized email from a Palm PDA or POP3 mail from another account.

The parameters are as follows:

- SenderDisplayName—The display name of the original sender. If this is NULL, no display name is added.
- SenderEmailAddress—The email address of the original sender. If this is NULL, no email address is added.
- SenderEmailAddressType—The type of email address. If NULL, no address type is added.
- CreationDate—The date the message was created. This value is reported as the actual creation date of the message, not the time of the call to AddExistingMessage.
- MessageBoxType—The type of message (see Section 4.30, “MessageBoxTypeConstants,” on page 184). If not specified (zero), it is a Draft message.
- MessageStatus—A bitmask that defines the status of the message (see Section 4.33, “MessageStatusConstants,” on page 185).
- MessagePriority—The priority of the message (see Section 4.31, “MessagePriorityConstants,” on page 185). If not set (zero) or set to an undefined value, egwNormal is assumed.
- MessageSecurity—The security of the message (see Section 4.34, “MessageSecurityConstants,” on page 186).
- DraftMessage (optional)—Points to a VARIANT structure of type VT_DISPATCH (all other types are ignored). A DIGWMessage pointer pointing to a Message object must be placed in the VARIANT data or an error is returned.
- LastModificationDate (optional)—Points to a VARIANT structure of type DATE (VT_EMPTY or VT_ERROR is ignored). This date is stored as the date and time of the last modification of the message. If this VARIANT is ignored or there is an error in obtaining the modification date, no modification date is stored for the message.

```
Message AddExistingMessageExt( String SenderDisplayname, String
SenderEmailAddress, String SenderEmailAddressType, Date Creation Date,
MessageBoxTypeConstants MsgBoxType, MessageStatusConstants MsgStatus,
MessagePriorityConstants MsgPriority, MessageSecurityConstants MsgSecurity,
[VARIANT DraftMsg], [VARIANT LastModificationDate], [VARIANT DeliveryDate])
```

Adds a message to the databse, but does not sent the message. The GroupWise sysytem treates these like InBox or OutBox items but does not try toi send them, This method is useful for synchronized email from a Palm, PDA, or POP3 mail from another account.

The parameters are as follows:

- SenderDisplayName—The display name of the original sender. If this is NULL, no display name is added.
- SenderEmailAddress—The email address of the original sender. If this is NULL, no email address is added.

- SenderEmailAddressType—The type of email address. If NULL, no address type is added.
- CreationDate—The date the message was created. This value is reported as the actual creation date of the message, not the time of the call to AddExistingMessage.
- MessageBoxType—The type of message (see Section 4.30, “MessageBoxTypeConstants,” on page 184). If not specified (zero), it is a Draft message.
- MessageStatus—A bitmask that defines the status of the message (see Section 4.33, “MessageStatusConstants,” on page 185).
- MessagePriority—The priority of the message (see Section 4.31, “MessagePriorityConstants,” on page 185). If not set (zero) or set to an undefined value, egwNormal is assumed.
- MessageSecurity—The security of the message (see Section 4.34, “MessageSecurityConstants,” on page 186).
- DraftMessage (optional)—Points to a VARIANT structure of type VT_DISPATCH (all other types are ignored). A DIGWMessage pointer pointing to a Message object must be placed in the VARIANT data or an error is returned.
- LastModificationDate (optional)—Points to a VARIANT structure of type DATE (VT_EMPTY or VT_ERROR is ignored). This date is stored as the date and time of the last modification of the message. If this VARIANT is ignored or there is an error in obtaining the modification date, no modification date is stored for the message.
- DeliveryDate (optional)—Points to a VARIANT structure of type DATE (VT_EMPTY or VT_ERROR is ignored). This date is stored as the date and time this message was delivered to the user. If this VARIANT is ignored or there is an error in abtaining the date, no delivery date is stored for the message.

NOTE: The AddExistingMessage and AddExistingMessageExt methods store a restore date on the message. This resotre date will prevent the new message from being auto purged or auto archived for 7 days. In addition, if the Smart purge feature is turned on, the restore date will prevent the message from being deleted from the Trash folder until it can be backed up or digested.

`MessageList Find(VARIANT Condition)`

Returns the MessageList collection containing the message matching the given Condition. Condition may be a string or a filter object. If condition is a string, it represents a filter expresion. See Chapter 5, “Filter Expressions,” on page 195. If Condition is a filter object, it represents a saved filter.

`Message Item(VARIANT Index)`

DEFAULT. Returns the Message specified by Index. Index may be a Long or a string. If Index is a Long, returns the Message object located at the given Index in the collection. Valid indexes are 1 through Count, inclusive. Throws an exception if the Index is outside of this range. If Index is a string, returns the Message object whose MessageID is equal to Index.

`Message Move(VARIANT Index, Messages Destination)`

Moves the message specified by Index from this collection to the collection specified by Destination. Index may be a Long, a string, or a Message object. If the Message object represented by Index and the Messages collection represented by Destination do not belong to the same account, an exception is thrown.

- If Index is a Long, it represents the Message object located at the given Index in the collection. Valid indexes are 1 through Count, inclusive. Throws an exception if the Index is outside of this range.

- If Index is a string, it represents the Message object whose MessageID is equal to Index.
- If Index is a Message object, it is the desired message.

When the Messages collection represents a shared folder, calling the Move method can cause a copy of the message to be added. That is, the returned message may be a copy of the message being added rather than simply returning the parameter passed in. This is because moving an item to a shared folder often results in the item being copied to a unique "prime" database. When this occurs, the original message (or link) is dissolved.

The Move method does not allow messages to be added to the Contacts folder or any Query folder. It allows messages with only a BoxType of egwOutgoing to be placed in the Sent Items systems folder.

The Move method allows any item to be placed into the Checklist folder. Placing any item in the Checklist folder gives the item the ability to have a Due Date, marks the item as Complete, and places the message in a specified spot.

`Remove(VARIANT Index)`

Removes the message specified by Index from this collection. Index may be a Long, a string, or a Message object. If Index is a Long, removes the Message object located at the given Index in the collection. Valid indexes are 1 through Count, inclusive.Throws an exception if the Index is outside of this range. If Index is a string, removes the Message object whose MessageID is equal to Index. If Index is a Message object, removes the given Message object from the collection.

`SequenceChecklistItem(ChecklistSequenceConstants MoveWhere, VARIANT vrMessageToSequence, VARIANT vrMessageBefore, vrMessageAfter)`

Moves the vrMessageToSequence item on the Master Checklist.

The three variant parameters: vrMessageToSequence, vrMessageBefore, and vrMessageAfter represent GroupWise Message objects. The three variants can be filled with a IGW_MESSAGE interface pointer, a BSTR holding the MessageID of the Message Object, or an integer with the Index into the Messages collection of the desired Message object.

If the MoveWhere constant is egwChecklistTop, the vrMessageToSequence is moved to the top of the Checklist. The vrMessageToSequence variant must hold a valid Checklist item. The vrMessageBefore and vrMessageAfter parameters are ignored.

If the MoveWhere constant specifies egwChecklist bottom, the vrMessageToSequence is moved to the bottom of the Checklist. The vrMessageBefore and vrMessageAfter parameters are ignored.

If the MoveWhere constant is egwChecklistBetween, the vrMessageToSequence is placed between the two messages represented by vrMessageBefore and vrMessageAfter. In this case, all three Variant parameters must point to valid Messages in the Checklist.

For the possible ChecklistSequence constants that are available, see Section 4.14, “ChecklistSequenceConstants,” on page 179.

Remarks

When the Messages collection represents a query folder, calling the Add method will throw an exception.

A Messages collection is refreshed when its parent object is refreshed. When a Messages collection is refreshed, it updates the Message objects in the collection but does not recursively refresh the Message objects themselves.

To add an existing message:

1. Add a Draft Message to the collection by calling Messages Collection::Add.
2. Populate the Draft Message with Recipients, body text, subject, and other data.
3. Call AddExistingMessage specifying the new Draft Message.

IMPORTANT: If successful, the Draft Message is placed in the Trash folder.

Note

Provides information and actions for a note. A subtype of Message.

Properties

The following table lists properties along with their access and descriptions.

Property	Access	Description
Accepted	R/O	Boolean. TRUE if this note has been accepted.
Archived	R/W	Boolean. TRUE if this message has been archived.
AutoDate	R/O	Boolean. TRUE if this note has an auto-date.
AutodateMessages	R/O	MessageList. The list of auto-dated items. This list includes the current note.
Categories	R/O	Categories Collection. The collection of Category objects on this Mail, Appointment, Document Reference, Note, Phone Message, Task, or AddressBookEntry.
ChecklistCompleted	R/W	Boolean. TRUE if the item is part of the Checklist folder Messages collection or a Task item AND the item has been marked Completed. FALSE otherwise.
ChecklistCompletedDate	R/W	Date. The date this item was marked completed.
ChecklistDueDate	R/W	Date. The date this item should be Completed. Returns an error if the Message item is not a Task or is not part of the Checklist folder Messages collection.
ChecklistPercentComplete	R/W	Integer. Between 0 and 100 percent, how complete the checklist item is. If this value is set to a number outside the 0 to 100 range, an error is returned. If the value is set to 100, the item will also be marked complete.
ChecklistShowOnCalendarDate	R/W	Date. The date this item should appear on the calendar task list.
CommonMessageID	R/O	String. The ID of the message, regardless of its location. For example, an email will have the same CommonMessageID for both Sent and Received.
Completed	R/W	Boolean. TRUE if this note has been completed.
DelayedDeliveryDate	R/W	Date. When the Message,Send() method is called, the message is not delivered until the specified Date. This property can be used only on a Draft Message.
Delegated	R/O	Boolean. TRUE if this note has been delegated.
DeliveredDate	R/O	Date. The date and time this message item was delivered to the account.
DraftAutoDates	R/O	DraftAutoDates collection. The collection contains additional Dates which the Appointment, Note, or Task is created on.
Forwarded	R/O	Boolean. TRUE if this item has been forwarded to another user. Otherwise, FALSE.

Property	Access	Description
MessageProperty	R/O	String. Based on the MessagePropertiesConstant passed in, a different string is returned.
NotifyWhenAccepted	R/W	Enum (NotifyMessageConstants). Specifies the type of notification to send when this note has been accepted.
NotifyWhenCompleted	R/W	Enum (NotifyMessageConstants). Specifies the type of notification to send when this note has been completed.
NotifyWhenDeclined	R/W	Enum (NotifyMessageConstants). Specifies the type of notification to send when this note has been deleted.
OnCalendar	R/W	Boolean. TRUE if this note should appear on a Calendar display.
Personal	R/O	Boolean. TRUE if this note is a personal note.
PersonalSubject	R/W	String. A text string to be shown by the GroupWise Windows client as the Subject text. This text appears only for the current user. The normal subject or their own Personal Subject text appears for other users.
Replied	R/O	Boolean. TRUE if this item has been replied to. Otherwise, FALSE.
Sharer	R/O	Address. The user who added this message to a shared folder. If this message was not added to a shared folder, this will be the same as the Sender property.
ShowOnChecklist	R/W	Boolean. TRUE if the item is part of the Checklist folder Messages collection. FALSE otherwise.
Size	R/O	Returns the size of the attachments and message body. If there are no attachments, only the size of the message body is returned.
StartDate	R/W	Date. The date a calendar note should start.

Methods

`Accept([String Comment], [Boolean AllInstances])`

Accepts this note. Comment is the text you would like to send in reply as you accept the note. AllInstances is used for auto-date notes. If AllInstances is set to TRUE, this method will accept all instances of the auto-dated note. Passing a value for AllInstances for a note that is not auto-dated is legal but ignored. If AllInstances is omitted, FALSE is assumed.

`Decline([String Comment], [Boolean AllInstances])`

Declines this note. Comment is the text you would like to send in reply as you decline the note. AllInstances is used for auto-date notes. If AllInstances is set to TRUE, this method will decline all instances of the auto-dated note. Passing a value for AllInstances for a note that is not auto-dated is legal but ignored. If AllInstances is omitted, FALSE is assumed.

`LocalDelete()`

Deletes this message only from the currently connected post office database. No synchronization will take place between the master database and a remote database.

`Note Delegate([Boolean AllInstances])`

Delegates this note. AllInstances is used for auto-date notes. If AllInstances is set to TRUE, this method will delegate all instances of the auto-dated note. Passing a value for AllInstances for a note that is not auto-dated is legal but ignored. If AllInstances is omitted, FALSE is assumed. Returns the OutBox note.

`RemoveChecklistDueDate()`

Removes the Checklist Due Date from the Message item. Returns an error if the item is not on the Checklist.

Remarks

When a Note object is refreshed, it updates the Message objects returned by its AutodateMessages property, but it does not recursively refresh the Message objects themselves.

PhoneMessage

A subtype of Message.

Properties

The following table lists properties along with their access and descriptions.

Property	Access	Description
Archived	R/O	Boolean. TRUE if this is an archive account.
CallerCompany	R/W	String. The caller's company name.
CallerName	R/W	String. The caller's name.
CameToSee	R/W	Boolean. TRUE if the person came to see you
Categories	R/O	Categories Collection. The collection of Category objects on this Mail, Appointment, Document Reference, Note, Phone Message, Task, or AddressBookEntry.
ChecklistCompleted	R/W	Boolean. TRUE if the item is part of the Checklist folder Messages collection or a Task item AND the item has been marked Completed. FALSE otherwise.
ChecklistCompletedDate	R/W	Date. The date this item was marked completed.
ChecklistDueDate	R/W	Date. The date this item should be Completed. Returns an error if the Message item is not a Task or is not part of the Checklist folder Messages collection.
ChecklistPercentComplete	R/W	Integer. Between 0 and 100 percent, how complete the checklist item is. If this value is set to a number outside the 0 to 100 range, an error is returned. If the value is set to 100, the item will also be marked complete.
ChecklistShowOnCalendarDate	R/W	Date. The date this item should appear on the calendar task list.
CommonMessageID	R/O	String. The ID of the message, regardless of its location. For example, an email will have the same CommonMessageID for both Sent and Received.
DelayedDeliveryDate	R/W	Date. When the Message.Send() method is called, the message is not delivered until the specified Date. This property can be used only on a Draft Message.
DeliveredDate	R/O	Date. The date and time this message item was delivered to the account.
DraftAutoDates	R/O	DraftAutoDates collection. The collection contains additional Dates that the appointment, note, or task is created on.
Forwarded	R/O	Boolean. TRUE if this item has been forwarded to another user. Otherwise, FALSE.
MessageProperty	R/O	String. Based on the MessagePropertiesConstant passed in, a different string is returned.

Property	Access	Description
PersonalSubject	R/W	String. A text string to be shown by the GroupWise Windows client as the Subject text. This text appears only for the current user. The normal subject or their own Personal Subject text appears for other users.
PhoneNumber	R/W	String. The caller's phone number.
PleaseCall	R/W	Boolean. TRUE if you are to return the caller's phone call.
Replied	R/O	Boolean. TRUE if this item has been replied to. Otherwise, FALSE.
ReturnedCall	R/W	Boolean. TRUE if the caller returned your phone call.
Sharer	R/O	Address. The user who added this message to a shared folder. If this message was not added to a shared folder, this will be the same as the Sender property.
ShowOnChecklist	R/W	Boolean. TRUE if the item is part of the Checklist folder Messages collection. FALSE otherwise.
Size	R/O	Returns the size of the attachments and message body. If there are no attachments, only the size of the message body is returned.
Telephoned	R/W	Boolean. TRUE if the person telephoned.
Urgent	R/W	Boolean. TRUE if this is an urgent message.
WantsToSee	R/W	Boolean. TRUE if the caller wants to see you.
WillCall	R/W	Boolean. TRUE if the caller will call you later.

Methods

See Message.

`LocalDelete()`

Deletes this message only from the currently connected post office database. No synchronization will take place between the master database and a remote database.

`RemoveChecklistDueDate()`

Removes the Checklist Due Date from the Message item. Returns an error if the item is not on the Checklist.

PhoneNumber

Describes a phone number in an address book entry. **--For GroupWise 2012 and later.**

Properties

The following table lists properties along with their access and descriptions.

Property	Access	Description
Application	R/O	Session. Returns the session containing this object.
Carrier	R/O	CellPhoneCarrierConstants. Returns the wireless carrier. Only valid for cell phone and car phone numbers.
DefailtPhoneNumber	R/O	Boolean. TRUE if this is the user's default phone number.
Parent	R/O	AddressBookEntry. Returns the owning address book entry.
PhoneNumber	R/O	String. The digits of the phone number
PhoneNumberType	R/O	PhoneNumberConstants. The type of the phone number.

Methods

`Delete()`

Deletes the phone number. Only valid on a personal address book entry.

`MakeDefaultPhoneNumber()`

Sets this as the default phone number for this contact. Only valid on a personal address book entry.

`GetCustomCarrier(string *bsCarrier, string *bsEmailGateway );`

Returns the carrier name and email gateway for this phone number if it is a custom defined carrier.

PhoneNumbers

A collection of all phone numbers defined for an address book entry. **--For GroupWise 2012 and later.**

Properties

The following table lists properties along with their access and descriptions.

Property	Access	Description
Application	R/O	Session. Returns the session containing this object.
Count	R/O	Returns the number of phone numbers.
Parent	R/O	AddressBookEntry. Returns the owning address book entry.

Methods

```
Item (long lIndex, PhoneNumber retval);
```

Returns the phone number at the given index (0 based).

```
ItemByType (PhoneNumberConstants PhoneNumberType, PhoneNumber **retval)
```

Returns the phone number of the given type if there. Otherwise returns an error.

```
Add( string NewPhoneNumber, PhoneNumberConstants NewPhoneNumberType,
DIGWPhoneNumber **retval )
```

Adds a new phone number to the collection. This only is allowed if the contact is in the personal address book.

Query

Provides query information and actions. This object can represent either a stand-alone query (which does not persist in the message database) or a query associated with a query folder (which does persist in the message database).

Properties

The following table lists properties along with their access and descriptions.

Property	Access	Description
Application	R/O	Application. The Application object.
Completed	R/O	Boolean. TRUE if the query has finished or has never before been started. FALSE if the query is still executing. When logged into a remote database, a running query will not complete until the user synchronizes (downloads) the query results.
CompletionDate	R/O	Date. The date and time the query was last completed, or if the query is still in progress, the date and time of the last update.
Expression	R/W	DEFAULT. String. The filter expression to be matched. This can be set to NULL or an empty string (""). In either case, no filtering will be applied to the query. See Chapter 5, “Filter Expressions,” on page 195.
Locations	R/O	Locations collection. The locations to be searched. Can contain Account, DocumentLibrary, and Folder objects.
Parent	R/O	Object. The Account object (for stand-alone queries) or Folder object (for query folders) that owns this object.
QueryMessages	R/O	MessageList. The messages that match the filter expression. Returns snapshots of the partial results if Completed = FALSE, or the final results if Completed = TRUE.
SearchLocally	R/W	Boolean. TRUE means execute the search locally, FALSE means execute the search on the master system (applies only when running Remote).
StartOnOpen	R/W	Boolean. TRUE means the client will automatically start the query when the query folder is opened. Affects only client behavior. Object API behavior is unaffected because folder objects are not opened.

Methods

`Folder CreateFolder(String FolderName, [Folder ParentFolder])`

Creates a new query folder based on this query. The new folder will have its own private query instance. FolderName is the name of the new folder. ParentFolder is the folder in which the new folder is created; if omitted, the new folder is created in the root folder. Returns the new folder.

`Refresh()`

Forces this Query object to reread property values from the message database.

- If this Query object belongs to a query folder, the folder’s Messages collection is not refreshed (consider refreshing the folder instead). Use this method to obtain a new snapshot while the query is executing.

- If the Query object belongs to a query folder, it is also refreshed when its owning Folder object is refreshed.
- If the Query object is a stand-alone query, it is not refreshed when its owning Account object is refreshed.

The actual reading of a specific property may be postponed until the next time the property is accessed. This "lazy evaluation" is an optimization that avoids unnecessary reading of unaccessed properties. Because Refresh may be used to update the Completed property, the reading of the QueryMessages property should be postponed. The CompletionDate and QueryMessages properties must be in sync (they must be read as simultaneously as possible).

`Start()`

Starts execution of the query. Sets Completed to FALSE and causes the QueryMessages property, or the parent folder's Messages collection, to start returning partial results.

`Stop()`

Stops execution of the query. It issues a HALT command to the GroupWise Search Engine. The HALT is handled the next time a set of Search Results is returned. The Stop method does not set the Query.Completed flag to TRUE. You can take the current set of found messaages and start using them.

Procedures

To perform a query:

1. Call the Account object's CreateQuery method to create a new Query object.
2. Set the Query's Expression, Location, and SearchLocally properties to the desired values.
3. Initiate the query by calling the Start method. Because queries can take some time, they execute asynchronously.
4. Poll the Completed property to determine when the query is done.

While the query is executing, QueryMessages returns snapshots of the partial results. (To examine the partial results, copy the QueryMessages collection to a variable and examine the variable, since QueryMessages returns a new snapshot each time it is accessed.) Once the query has completed, QueryMessages contains the final results.

To create a query folder:

1. Create a stand-alone Query object as described above, or access the Query object in an existing query folder.
2. Call the Query object's CreateFolder method to create the new query folder.

The query folder's query can be re-executed by calling Folder.Query.Start( ). The results, both partial and final, appear in Folder.Messages (Folder.Query.QueryMessages is always empty).

Remarks

When a Query object is refreshed, it recursively refreshes the Locations collection returned by its Location property. It also updates the Message objects by its QueryMessages property, but it does not recursively refresh the Messages objects themselves.

QuickMessage

Provides a faster way to access messages.

Properties

The following table lists properties along with their access and descriptions.

Property	Access	Description
Accepted	R/O	Boolean. TRUE if the QuickMessage is an Appointment, Task, or Note item and has been accepted by the user. If the item isn't an Appointment, Task, or Note item, S_FALSE is also returned (in addition to FALSE).
Application	R/O	Application. Returns the application object.
BoxType	R/O	MessageBoxTypeConstants. Returns an enumeration value for Message that corresponds to whether the item is one of the following: Received egwIncoming Sent egwOutgoing Personal egwPersonal Draft egwDraft
Categories	R/O	Categories. Returns the Categories collection object that corresponds to the QuickMessage. Any changes made to the Categories or Category objects that are accessed from QuickMessage are not reflected back to the QuickMessage object. If changes are made, the Categories object is released, and the Categories object is retrieved again, the old Categories object is returned.
EnclosingFolders	R/O	Folders. Returns a Folders collection object that holds every folder this QuickMessage appears in.
Hidden	R/O	Boolean. TRUE if this QuickMessage is hidden. If a QuickMessage is hidden, it does not show up in any user interface.
Message	R/O	Message. Returns the full Message object that corresponds to this QuickMessage. Any changes that are made to the Message object (or any of its sub objects) are not given to the QuickMessage object. To see the changes, the entire QuickMessages collection and the associated QuickMessage objects must be released and reacquired.
ModifiedDate	R/O	Date. The date and time the QuickMessage was last modified.
Opened	R/O	Boolean. TRUE if this QuickMessage has been opened.
Parent	R/O	QuickMessages. Returns the QuickMessages collection object that spawned the QuickMessage object.
PersonalSubject	R/O	String. A text string that is displayed by the GroupWise Windows client as the subject text. The text appears only for the current user. The normal subject or their own personal subject text appears for other users.
Private	R/O	Boolean. TRUE if the QuickMessage is marked Private (so that a proxy user cannot see it).
Read	R/O	Boolean. TRUE if the QuickMessage has been read.
Subject	R/O	String. The subject of the message. Unlike the full Message object, QuickMessage returns only the plain text version of the subject.

Remarks

A QuickMessage is a small subset of the full Message object. QuickMessage properties should match their corresponding Message object. However, if the full Message object is obtained and then changed, the changes do not make it back into the QuickMessage object. To refresh QuickMessage, the entire QuickMessages collection and all related objects from the collection must be released and then retrieved again.

QuickMessages

Provides a fast way to access messages.

Properties

The following table lists properties along with their access and descriptions.

Property	Access	Description
Application	R/O	Application. Returns the application object.
ClassName	R/O	Returns the ClassName for the Message. For example, a Mail Message returns the string GW.MESSAGE.MAIL.
Count	R/O	Long. The number of objects in this collection.
_NewEnum	R/O	Enumeration object. Implements IEnumVARIANT. For Windows.
Parent	R/O	Object. The IDispatch pointer to the object that owns the collection.
StartDate	R/O	Returns the Date a Calendar Item starts on. For example, the starting time of an appointment. Only Calendar Items (Appointments, Tasks, and Reminder Notes) return a date. If you request the StartDate from any other Message Item type, an error is returned.

Methods

`QuickMessage Item(Long Index)`

DEFAULT. Returns the QuickMessage object that is located at the given Index in the collection. Index can only be a Long numeric value from 1 to the number returned from the Count property. Returns an exception if the Index is outside that range.

Recipient

Describes the recipient of a message.

Properties

The following table lists properties along with their access and descriptions.

Property	Access	Description
Address	R/O	Address. The address this recipient resolved to. Can be Nothing if this recipient is unresolved or resolved as an external address.
Application	R/O	Application. The Application object.
DisplayName	R/W	String. A descriptive name to display to users.
EmailAddress	R/W	DEFAULT. String. The email address used by the system to deliver mail. The format is determined by the EmailType property.
EmailType	R/W	String. The type of email address. If you want GroupWise to transport email, you must use the NGW type to add any type of recipient. Non-GroupWise addresses are submitted to the operating system's default email transport. (In Windows, for example, the transport would be MAPI.) For examples of how to call Mail.Recipients.Add, see the Remarks section of Recipients (page 150).
Parent	R/O	Recipients collection. The Recipients collection that owns this object.
RecipientStatusList	R/O	RecipientStatusList collection object.
Resolved	R/W	Enum (ResolvedConstants). Indicates the resolved status of this recipient. Automatically set to egwNotResolved when the DisplayName, EmailAddress, EmailType, or TargetType properties are changed. Can be manually set only to egwNotResolved. This property is not persistent. It will revert to egwNotResolved whenever the Recipient object is refreshed or freed from memory.
TargetType	R/W	Enum (AddressTargetTypeConstants). Indicates whether or not this recipient is a primary recipient, carbon copy recipient, or blind copy recipient.

Methods

`Delete()`

Removes this recipient from the owning Recipients collection and deletes the Recipient object.

`Resolve([AddressBook ResolveTo])`

Resolves this recipient. See Remarks for more information.

Remarks

A Recipient object differs from an Address object in that it originates from an arbitrary address and has a TargetType and a Resolve method. In contrast, an Address object originates from a resolved address and does not have a TargetType or a Resolve method.

A Recipient object is refreshed when its parent object is refreshed. When a Recipient object is refreshed, it recursively refreshes its Address object.

The Resolve method behaves in the following manner:

- If the Recipient is already resolved (its Resolved property is TRUE), the Resolve operation is considered successful.
- Otherwise, when attempting to locate the indicated address, three address books are searched in the following order: (a) Frequent Contacts personal address book; (b) default address book; (c) system address book. This is the same search order used by the Name Completion Control when looking up names in the GroupWise client.
- If an entry is found that matches the Recipient fields indicated by the caller, a new Address property is created for the Recipient and its values are copied from the found address book entry. The Resolve operation is considered successful.
- If the recipient is not found in an address book and its EmailType is external (such as an Internet address), the Recipients Address property is set to a new Address object with ObjType = egwUser and DisplayName, EmailAddress, and EmailType the same as the Recipient's properties, the Resolve operation is considered successful.
- If the recipient is not found in an address book and its EmailType is internal (such as a GroupWise address), the Resolve operation throws an exception.

Recipients

A collection of Recipient objects.

Properties

The following table lists properties along with their access and descriptions.

Property	Access	Description
Application	R/O	Application. The Application object.
Count	R/O	Long. The number of objects in this collection.
_NewEnum	R/O	Enumeration object. Implements IEnumVARIANT. For Windows only.
Parent	R/O	Message. The Message object that owns this collection.

Methods

`Recipient Add(Address Address, [AddressTargetTypeConstants TargetType])`

Creates an unresolved Recipient object from an Address object, then adds the Recipient object to the collection. The new Recipient's Address property is set to the Address object given, and its DisplayName, EmailAddress, and EmailType are copied from the Address object. If TargetType is omitted, egwTo is assumed. Returns the new Recipient. See Remarks below.

`Recipient Add(Addresses Addresses, [AddressTargetTypeConstants TargetType])`

Creates unresolved Recipients for each Address in the Addresses collection given, and adds each Recipient object to the collection. If TargetType is omitted, egwTo is assumed. Returns nothing. See Remarks below.

`Recipient Add(String EmailAddress, [String EmailType], [AddressTargetTypeConstants TargetType])`

Creates an unresolved Recipient from the EmailAddress, EmailType, and TargetType, then adds the Recipient object to the collection. If EmailType is omitted, an empty string ("") is assumed. If TargetType is omitted, egwTo is assumed. The new Recipient's Address property is set to Nothing and its DisplayName is set to EmailAddress. Returns the new Recipient. See Remarks below. See also AddressTargetTypeConstants.

`Recipient AddByDisplayName(String DisplayName, [AddressTargetTypeConstants TargetType])`

Creates an unresolved Recipient from DisplayName, then adds it to the collection. If DisplayName is an empty string (""), an exception is thrown. If TargetType is omitted, To is assumed. The new Recipient Address property is set to Nothing and its EmailAddress and EmailType are set to an empty string (""). Returns the new Recipient. See AddressTargetTypeConstants.

`Recipient Item(Long Index)`

DEFAULT. Returns the Recipient object located at the given Index in the collection. Valid indexes are 1 through Count inclusive. Throws an exception if the Index is outside of this range.

Resolve([AddressBook ResolveTo])

Performs a Resolve on each Recipient in the collection. If an error occurs, it leaves that recipient unresolved and proceeds to the next recipient. Throws an exception if any recipients failed to resolve. Check the Resolved property of each Recipient object to determine which ones failed.

Remarks

The Add method template is as follows:

Add(VARIANT P1, [VARIANT P2], [VARIANT P3])

The Add method checks the P1 type at runtime.

- If P1 is an Address object, the first form of Add is assumed; P2, if present, must be a TargetType Enum of type AddressTargetTypeConstants, and P3 must not be present.
- If P1 is an Addresses collection, the second form of Add is assumed; P2, if present, must be a TargetType Enum of type AddressTargetTypeConstants, and P3 must not be present.
- If P1 is a String, the third form of Add is assumed; P2, if present, must be a string, and P3, if present, must be a Target Enum of type AddressTargetTypeConstants.

For example, you can use either of the following examples to Add a recipient:

```
Mail.Recipients.Add('devsup@novell.com', 'NGW', egwTo);
Mail.Recipients.Add('devsup@novell.com', 'NGW');
```

AddByDisplayName must be a separate method from Add, since EmailAddress and DisplayName are both Strings and cannot be distinguished by type.

A Recipients collection is refreshed when its parent object is refreshed. When a Recipients collection is refreshed, it recursively refreshes its contained Recipient objects.

RecipientStatus

Describes the status of a recipient.

Properties

The following table lists properties along with their access and descriptions.

Property	Access	Description
Application	R/O	Application. The Application object.
Comment	R/O	String. Any comment associated with this action. Most comments are added when messages are accepted or declined.
Parent	R/O	Recipient. The parent recipient object.
StatusDate	R/O	Date. The day and time this status occured.
StatusType	R/O	RecipientStatusConstants. Returns the type of status (action) that occured for this recipient.

Methods

There are no methods for this object.

RecipientStatusList

Recipient status list.

Properties

The following table lists properties along with their access and descriptions.

Property	Access	Description
Application	R/O	Application. The application object.
Count	R/O	Long. The count of RecipientStatus objects available.
Parent	R/O	Recipient. The parent recipient object.

Methods

`RecipientStatus Item(Long lIndex)`

Default. Returns the RecipientStatus object located at the given index in the collection. Valid indexes are 1 through Count, inclusive. Throws an exception if the index is outside of this range.

Rule

A rule object defines a rule in the account.

Properties

The following table lists properties along with their access and descriptions.

Property	Access	Description
Actions	R/O	RuleActions. The collection of rule actions.
Application	R/O	Application. The Application object.
Appointment	R/W	Boolean. The rule will trigger on the type being an appointment.
Conflict	R/W	AppointmentConflictConstants. Specifies the trigger of rules whether or not there is a conflicting appointment at the same time as the new appointment.
Deleted	R/O	Boolean. Whether or not the rule is deleted.
Draft	R/W	Boolean. The rule triggers on the BoxType being a Draft item.
Enabled	R/W	Boolean. Whether or not the rule is enabled.
Event	R/W	RuleContents. The rule event type.
Filter	R/W	String. The filter for the rule.
Folder	R/W	Folder. The folder involved in the rule.
Mail	R/W	Boolean. The rule triggers on the type being a mail item.
Name	R/W	String. The name of the rule.
Note	R/W	Boolean. The rule triggers on the type being a note.
Parent	R/O	Rules Collection. The Rules collection the owns this rule.
Phone	R/W	Boolean. The rule triggers on the type being a phone message item.
Posted	R/W	Boolean. The rule triggers on the BoxType being a Posted/personal item.
Received	R/W	Boolean. The rule triggers on the BoxType being a Received item.
Sequence	R/W	Long. The order of this rule in the list of rules.
Sent	R/W	Boolean. The rule triggers on the BoxType being a Sent item.
Task	R/W	Boolean. The rule triggers on the type being a task.

Methods

`Delete()`

Deletes this rule from its parent collection.

`Refresh()`

Reread the Rule object.

RuleAction

A rule action object defines a rule action in the rule.

Properties

The following table lists properties along with their access and descriptions.

Property	Access	Description
BC	R/W	String. Users to include in the BC line of the item used in the rule.
BusyType	R/W	TypeBlockConstants. Specifies busy type to set in the item used in the rule.
Categories	R/O	Categories Collection. The collection of Category objects on item used in the rule.
CC	R/W	String. Users to include in the CC line of the item used in the rule.
Comment	R/W	String. Comment string in an accept or decline.
CommentToSender	R/W	String. Comment string in an accept or decline.
Folder	R/W	Folder. The folder involved in the rule action.
From	R/W	String. From user included in item used in rule.
Message	R/W	String. Message body included in item used in rule.
Parent	R/O	Rule. The Rule that owns this rule.
Subject	R/W	String. Subject included in item used in the rule.
To	R/W	String. Users to include in the To line of the item used in the rule.
Type	R/W	RuleActionConstants. Specifies the type.

Methods

`Delete()`

Deletes this rule from its parent collection.

RuleActions

The RulesActions collection holdes the RuleAction objects for a Rule.

Properties

The following table lists properties along with their access and descriptions.

Property	Access	Description
Count	R/O	Long. The number of objects in this collection.
_NewEnum	R/O	Enumeration object. Implements IEnumVARIANT. For Windows only.
Parent	R/O	Account. The Account object that owns this collection.

Methods

`RuleAction Add(RuleActionConstants type)`

Creates a new rule action with the given type.

`Commit()`

Commit the changes to the RuleAction.

`RuleAction Item(Long Index)`

DEFAULT. Returns the RuleAction object specified by Index. Valid indexes are 1 through Count, inclusive.

Rules

The Rules collection holds the Rule object.

Properties

The following table lists properties along with their access and descriptions.

Property	Access	Description
Application	R/O	Application. The application object.
Count	R/O	Long. The number of objects in this collection.
_NewEnum	R/O	Enumeration object. Implements IEnumVARIANT. For Windows only.
Parent	R/O	Account. The Account object that owns this collection.

Methods

`Rule Add(String Name, Rule Constants event, [VARIANT] sequence)`

Creates a new rule with the given Name. The rule event must be specified. Optionally you can specify the sequence order of the rule in the collection of rules.

`Rule Item(Long Index)`

DEFAULT. Returns the Rule object specified by Index. Valid indexes are 1 through Count, inclusive.

SharedNotification

Notifies a user of a shared folder or a shared personal address book. A subtype of Message.

Properties

The following table lists properties along with their access and descriptions.

Property	Access	Description
Archived	R/W	Boolean. TRUE if this message has been archived.
Categories	R/O	Categories Collection. The collection of Category objects on this Mail, Appointment, Document Reference, Note, Phone Message, Task, or AddressBookEntry.
ChecklistCompleted	R/W	Boolean. TRUE if the item is part of the Checklist folder Messages collection or a Task item AND the item has been marked Completed. FALSE otherwise.
ChecklistCompletedDate	R/W	Date. The date this item was marked completed.
ChecklistDueDate	R/W	Date. The date this item should be Completed. Returns an error if the Message item is not a Task or is not part of the Checklist folder Messages collection.
ChecklistPercentComplete	R/W	Integer. Between 0 and 100 percent, how complete the checklist item is. If this value is set to a number outside the 0 to 100 range, an error is returned. If the value is set to 100, the item will also be marked complete.
ChecklistShowOnCalendarDate	R/W	Date. The date this item should appear on the calendar task list.
CommonMessageID	R/O	String. The ID of the message, regardless of its location. For example, an email will have the same CommonMessageID for both Sent and Received.
DeliveredDate	R/O	Date. The date and time this message item was delivered to the account.
Description	R/O	String. The description given to this notification by the originator.
Forwarded	R/O	Boolean. TRUE if this item has been forwarded to another user. Otherwise, FALSE.
MessageProperty	R/O	String. Based on the MessagePropertiesConstant passed in, a different string is returned.
Name	R/O	String. The name assigned to the notification by the originator.
NotificationType	R/O	Enum (SharedNotificationTypeConstants) The type of notification.
PersonalSubject	R/W	String. A text string to be shown by the GroupWise Windows client as the Subject text. This text appears only for the current user. The normal subject or their own Personal Subject text appears for other users.
Replied	R/O	Boolean. TRUE if this item has been replied to. Otherwise, FALSE.

Property	Access	Description
Sharer	R/O	Address. The user who added this message to a shared folder. If this message was not added to a shared folder, this will be the same as the Sender property.
ShowOnChecklist	R/W	Boolean. TRUE if the item is part of the Checklist folder Messages collection. FALSE otherwise.
Size	R/O	Returns the size of the attachments and message body. If there are no attachments, only the size of the message body is returned.

Methods

`Accept([Folders Folders], [String FolderName], [String Description])`

Accepts a shared folder or shared personal address book. Folders is the Folders collection to which you wish to add this folder. Name is the name you wish to give to this folder. Description is the description you wish to give to this folder. Any parameters not specified are defaulted from the original object.

`Accept([String SharedAddressBookName], [String Description])`

Accepts a shared personal address book. SharedAddressBookName is the name you wish to call this address book. Description is the description you wish to give to this address book. Any parameters not specified are defaulted from the original object.

`LocalDelete()`

Deletes this message only from the currently connected post office database. No synchronization will take place between the master database and a remote database.

`RemoveChecklistDueDate()`

Removes the Checklist Due Date from the Message item. Returns an error if the item is not on the Checklist.

Signature

Describes a Signature object that can be added to an email message.

Properties

The following table lists properties, along with their access and descriptions.

Property	Access	Description
Application	R/O	Application. The Application object.
Body	R/W	Safe Array. The bytes that define an HTML signature.
Global Signature	R/O	Boolean. TRUE if this is a global signature; otherwise FALSE.
Name	R/W	String. The name of this Signature.
Parent	R/O	Signatures. The Signatures object that owns this Signature object.
Type	T/O	Integer. The type of Signature. For possible values, see the Remarks section.

Methods

`Default()`

Sets this Signature object as the default signature to be added to outgoing email messages.

`Delete()`

Deletes this Signature object from the GroupWise system.

Remarks

The Type property can have the following values:

Value	Constant	Description
0	egwUnknownSignature	Unknown.
1	egwHtmlSignature	HTML text contained in the Signature bytes.
2	egwTextSignature	Plain (ANSI) text contained in the Signature bytes.

Signatures

Contains the Signature objects for a GroupWise Account object. This object is usually retrieved by calling the get_Signatures method of the Account object.

Properties

The following table lists properties, along with their access and descriptions.

Property	Access	Description
Application	R/O	Application. The Application object.
Count	R/O	Long. The number of objects in this collection.
Parent	R/O	Account. The Account object that owns this Signatures collection.

Methods

`Signature Add(String Name, VARIANT vrFileNameOrSafeArray)`

Adds a Signature with the specified name to the database. If the Name is empty or if the Name is already in the collection, an error is returned.

The vrFileNameOrSafeArray must contain either a string or a safe array (of type VT_ARRAY) or an error is returned. If the VARIANT is a string, it must point to a file on disk that contains the Signature data. If the VARIANT contains a safe array, it must point to an array of HTML bytes that define the Signature. The HTML becomes the Signature that is added to an email message just before it is sent.

`Signature Item(VARIANT vrIndex)`

(Default) Returns a Signature object that matches the specified vrIndex. The index must be of type String or Numeric or an error is returned. If it is a string, the system searches the Signatures collection for a Signature with a name that matches the given string. If a matching name is not found, the system returns a NULL pointer to the Signature object and doesn't return an error. If the index contains a number, the signature at that position in the collection is returned.

Signature collections are numbered from 1 to the number of Signatures. An error is returned for any number outside of this boundary.

`Signature GetDefault()`

Returns the default Signature object for the Account.

Task

Provides information and actions for a task. A subtype of Message.

Properties

The following table lists properties along with their access and descriptions.

Property	Access	Description
Accepted	R/O	Boolean. TRUE if this task has been accepted.
Archived	R/W	Boolean. TRUE if this message has been archived.
AssignedDate	R/W	Date. The date and time to which this task is assigned. 0 = Master Task List item.
Autodate	R/O	Boolean. TRUE if this task has an auto-date.
AutodateMessages	R/O	MessageList. List includes current item.
Categories	R/O	Categories Collection. The collection of Category objects on this Mail, Appointment, Document Reference, Note, Phone Message, Task, or AddressBookEntry.
ChecklistCompleted	R/W	Boolean. TRUE if the item is part of the Checklist folder Messages collection or a Task item AND the item has been marked Completed. FALSE otherwise.
ChecklistCompletedDate	R/O	Date. The date this item was marked complete.
ChecklistDueDate	R/W	Date. The date this item should be Completed. Returns an error if the Message item is not a Task or is not part of the Checklist folder Messages collection.
ChecklistPercentComplete	R/W	Integer. Between 0 and 100 percent, how complete the checklist item is. If this value is set to a number outside the 0 to 100 range, an error is returned. If the value is set to 100, the item will also be marked complete.
ChecklisShowOnCalendarDate	R/W	Date. The date this item should appear on the calendar task list.
CommonMessageID	R/O	String. The ID of the message, regardless of its location. For example, an email will have the same CommonMessageID for both Sent and Received.
Completed	R/W	Boolean. TRUE if this task has been completed.
DelayedDeliveryDate	R/W	Date. When the Message.Send() method is called, the message is not delivered until the specified Date. This property can be used only on a Draft Message.
Delegated	R/O	Boolean. TRUE if this task has been delegated.
DeliveredDate	R/O	Date. The date and time this message item was delivered to the account.
DraftAutoDates	R/O	DraftAutoDates collection. The collection contains additional Dates which the Appointment, Note, or Task is created on.
DueDate	R/W	Date. The date and time this task is due to be completed.

Property	Access	Description
Forwarded	R/O	Boolean. TRUE if this item has been forwarded to another user. Otherwise, FALSE.
MessageProperty	R/O	String. Based on the MessagePropertiesConstant passed in, a different string is returned.
NotifyWhenAccepted	R/W	Enum (NotifyMessageConstants). Specifies the type of notification to send when this task has been accepted.
NotifyWhenCompleted	R/W	Enum (NotifyMessageConstants). Specifies the type of notification to send when this task has been completed.
NotifyWhenDeclined	R/W	Enum (NotifyMessageConstants). Specifies the type of notification to send when this task has been declined.
OnCalendar	R/W	Boolean. TRUE if this task, should appear on a Calendar display.
Personal	R/O	Boolean. TRUE if this task is a personal task.
PersonalSubject	R/W	String. A text string to be shown by the GroupWise Windows client as the Subject text. This text appears only for the current user. The normal subject or their own Personal Subject text appears for other users.
Replied	R/O	Boolean. TRUE if this item has been replied to. Otherwise, FALSE.
Sharer	R/O	Address. The user who added this message to a shared folder. If this message was not added to a shared folder, this will be the same as the Sender property.
ShowOnChecklist	R/W	Boolean. TRUE if the item is part of the Checklist folder Messages collection. FALSE otherwise.
Size	R/O	Returns the size of the attachments and message body. If there are no attachments, only the size of the message body is returned.
StartDate	R/W	Date. The date and time this task is to start.
TaskCategory	R/W	String. Assigning a longer string causes an exception to be thrown.
TaskCompletedDate	R/O	Date. The date this item was marked complete.
TaskPercentComplete	R/W	Integer. Between 0 and 100 percent, how complete the checklist item is. If this value is set to a number outside the 0 to 100 range, an error is returned. If the value is set to 100, the item will also be marked complete.
TaskPriority	R/W	Long. The numerical priority of this task within its category.

Methods

`Accept([String Comment], [Boolean AllInstances])`

Accepts this task. Comment is the text you would like to send in reply as you accept the task. AllInstances is used for auto-date tasks. If AllInstances is set to TRUE, this method will accept all instances of the auto-dated task. Passing a value for AllInstances for a task that is not auto-dated is legal but ignored. If AllInstances is omitted, FALSE is assumed.

`Appointment Delegate([Boolean AllInstances])`

Delegate this task. AllInstances is used for auto-date tasks. If AllInstances is set to TRUE, this method will delegate all instances of the auto-dated task. Passing a value for AllInstances for a task that is not auto-dated is legal but ignored. If AllInstances is omitted, FALSE is assumed.

`Decline([String Comment], [Boolean AllInstances])`

Declines this task. Comment is the text you would like to send in reply as you decline the appointment. AllInstances is used for auto-date tasks. If AllInstances is set to TRUE, this method will decline all instances of the auto-dated task. Passing a value for AllInstances for a task that is not auto-dated is legal but ignored. If AllInstances is omitted, FALSE is assumed.

`LocalDelete()`

Deletes this message only from the currently connected post office database. No synchronization will take place between the master database and a remote database.

`MoveToMasterTaskList()`

Move this task to the master task list.

`RemoveChecklistDueDate()`

Removes the Checklist Due Date from the Message item. Returns an error if the item is not on the Checklist.

Remarks

When a Task object is refreshed, it updates the Message objects returned by its AutodateMessages property, but it does not recursively refresh the Message objects themselves.

TimeBlock

Describes a block of time as part of a search for busy/free time.

Properties

The following table lists properties along with their access and descriptions.

Property	Access	Description
Application	R/O	Application. The Application object.
Duration	R/O	Double. The duration, in days, of the block of time. For example, 0.5 = 12 hours, 1/24 = 1 hour, etc. Gives accuracy to microseconds.
EndDate	R/O	Date. The ending date and time of this time block.
From	R/O	Address. The sender of the appointment. Nothing if ObjType = egwFree or if appointment information is unavailable to the user.
ObjType	R/O	Enum (TimeBlockConstants). The type of this block of time.
Parent	R/O	TimeBlocks collection. The TimeBlocks collection that owns this object.
Place	R/O	String. The place of the appointment. An empty string ("") if ObjType = egwFree or if appointment information is unavailable to the user.
StartDate	R/O	Date. The starting date and time of this time block.
Subject	R/O	String. The subject of the appointment that is occupying this time block. An empty string ("") if ObjType = egwFree or if appointment information is unavailable to the user.

Remarks

A TimeBlock object is refreshed when its parent object is refreshed. When a TimeBlock is refreshed, it recursively refreshes the Address object returned by its From property.

TimeBlocks

A collection of TimeBlock objects.

Properties

The following table lists properties along with their access and descriptions.

Property	Access	Description
Application	R/O	Application. The Application object.
Count	R/O	Long. The number of objects in this collection.
_NewEnum	R/O	Enumeration object. Implements IEnumVARIANT. For Windows only.
Parent	R/O	BusySearchElement. The BusySearchElement object that own this collection.

Methods

`TimeBlock Item(Long Index)`

DEFAULT. Returns the TimeBlock object located at the given Index in the collection. Valid indexes are 1 through Count inclusive. Throws an exception if the Index is outside of this range.

Remarks

If the first element in the collection is a Free TimeBlock, its start time will be identical to the Start Time shown in Preferences. However, the time fields associated with a non-Free TimeBlock always indicate the actual Start and End times of the associated appointment.

A TimeBlocks collection object is refreshed when its parent object is refreshed. When a TimeBlocks collection is refreshed, it recursively refreshes its contained TimeBlock objects.

Trash

An account's Trash.

Properties

The following table lists properties along with their access and descriptions.

Property	Access	Description
Application	R/O	Application. The Application object.
Name	R/O	String. The name of this trash folder.
Parent	R/O	Account. The Account object that owns this object.
TrashEntries	R/O	TrashEntries collection. The collection of entries in the trash.

Methods

`Empty()`

Empties the Trash by deleting all entries from the TrashEntries collection.

`Refresh()`

Forces this Trash object and its associated TrashEntries collection to reread property values from the message database. The reading of a specific property may be postponed until the next time the property is accessed. This "lazy evaluation" is an optimization that avoids unnecessary reading of unaccessed properties.

TrashEntries

A collection of TrashEntry objects in the Trash object, or an independent standalone list of Trash entries.

Properties

The following table lists properties along with their access and descriptions.

Property	Access	Description
Application	R/O	Application. The Application object.
Count	R/O	Long. The number of objects in this collection.
_NewEnum	R/O	Enumeration object. Implements IEnumVARIANT. For Windows only.
Parent	R/O	Object. The Trash object (for the contents of the trash) or the Application object (for a stand-alone collection returned from Find) that owns this collection.

Methods

`TrashEntries Find(VARIANT Condition)`

Returns a new collection that contains the TrashEntry objects whose Message properties match the filter expression given by the Condition parameter. Condition may be either a string or a Filter object. If Condition is a string, it represents a filter expression. See Chapter 5, “Filter Expressions,” on page 195. If Condition is a Filter object, it represents a saved filter.

`TrashEntry Item(Long Index)`

DEFAULT. Returns the TrashEntry object located at the given Index in the collection. Valid indexes are 1 through Count inclusive. Throws an exception if the Index is outside of this range.

`TrashEntry ItemEx(VARIANT Index)`

Returns the TrashEntry object located specified by Index. Index may be a string, a Long, or a Message Object. If Index is a string, retuns the TrashEntry object whose Message object’s MessageID property is equal to Index. If Index is a Long, retuns the TrashEntry object located at the given Index in the collection. Valid indexes are 1 through Count inclusive. Throws an exception if the Index is outside of this range. If Index is a Message object, returns the TrashEntry object whose Message property is equal to Index.

TrashEntry

A Trash entry.

Properties

The following table lists properties along with their access and descriptions.

Property	Access	Description
Application	R/O	Application. The Application object.
Folder	R/O	Folder. The original folder that contained this item.
Message	R/O	Message. The Message object for this item.
Parent	R/O	TrashEntries. The TrashEntries collection that owns this object.

Methods

`Delete()`

Removes this TrashEntry object from its owning TrashEntries collection and deletes this object.

`Undelete()`

Removes this TrashEntry object from its owning TrashEntries collection, deletes this object, and moves its Message object back to the folders from which it was deleted. If the folders that contained it no longer exist, the Message object is placed into the Cabinet folder.

Remarks

A TrashEntry object is automatically created when a message is deleted from a folder. A TrashEntry object is automatically deleted if the message is moved back to a folder from which it was deleted.

4 Constants

The Object API constants provide a text string that is associated with each numerical value, as shown in the following sections:

4.1 AccountPropertiesConstants

These constants identify the type of string that is returned from calling the AccountProperty of Account (page 26).

Constant	Value	Description
egwAccountPropertyDomain	0x0001	The domain for the user.
egwAccountPropertyPostOffice	0x0002	The Post Office for the user.
egwAccountPOAStillAlive	0x0003	Whether the POA is responding to requests.
egwAccountLastError	0x0004	The last GroupWise engine error code.
egwAccountSpaceUsedMB	0x0006	In Megabytes, the amount of disk space used by this account.
egwAccountSpaceLimitMB	0x0007	In Megabytes, the limit on disk space that can be used by this account.
egwAccountSpaceThreshhold	0x0008	As a percentage, how close to the disk space limit this account’s actual disk space used is .
egwAccountFID	0x0009	Return the GroupWise FID for this account.
egwAccountUID	0x000A	Return the unique ID for the logged in user.

4.2 AccountRightsConstants

These constants identify the type of access rights given to a user for an account.

Constant	Value	Description
egwCanArchive	1	The user can archive messages.
egwPrefsRulesGroups	2	The user can modify preferences, rules, and groups.
egwReadAppointments	4	The user can read appointments.
egwReadMailAndPhone	8	The user can read mail and phone messages.
egwReadNotes	16	The user can read notes.
egwReadPrivate	32	The user can read private messages.
egwReadTasks	64	The user can read tasks.
egwReceiveAlarms	128	The user can receive alarms.
egwReceiveNotifications	256	The user can receive notifications.
egwWriteAppointments	512	The user can write appointments.
egwWriteMailAndPhone	1024	The user can write mail and phone messages.
egwWriteNotes	2048	The user can write notes.
egwWriteTasks	4096	The user can write tasks.

4.3 AddressBookRightsConstants

These constants identify the access rights a user has to an address book.

Constant	Value	Description
egwReadOnly	0	Read-only access.
egwFullAccess	1	Read/write access.

4.4 AddressBookSubtypeConstants

These constants identify the subtype of an address book. See also AddressBookTypeConstants.

Constant	Value	Description
egwOther	0	Any address book other than the Novell Frequent Contacts personal address book.
egwFrequentContact	1	Novell Frequent Contacts personal address book.

4.5 AddressBookTypeConstants

These constants identify the type of an address book. See also AddressBookSubtypeConstants.

Constant	Value	Description
egwUnknownBook	0	Unknown address book.
egwNovellPersonal	1	Novell personal address book.
egwNovellSystem	2	Novell system address book.

4.6 AddressTargetTypeConstants

These constants identify the target type (recipient type) of an address.

Constant	Value	Description
egwTo	0	To. A primary recipient.
egwCC	1	Carbon Copy. A secondary recipient.
egwBC	2	Blind Copy. This address is hidden to all users except the sender and the receiver.

4.7 AddressTypeConstants

These constants identify the type of an address.

Constant	Value	Description
egwUnknown	0	Unknown.
egwUser	1	User.
egwCompany	2	Company.
egwResource	3	Resource, such as a conference room.
egwGroup	4	Group.

4.8 AppointmentConflictConstants

These constants identify how to trigger on an appointment received.

Constant	Value	Description
egwNo	0	No conflicting appointment.
egwYes	1	A conflicting appointment.
egwDontCare	-1	Don’t care if there is a conflicting appointment.

4.9 AttachmentTypeConstants

These constants identify the type of a message attachment

Constant	Value	Description
egwUnknownAttachment	0	Unknown.
egwFile	1	File.
egwMessage	2	Embedded message.
egwMovie	3	Multimedia object.
egwOLEEmbedded	4	Embedded OLE object.
egwOLELinked	5	Linked OLE object.
egwSoundAnnotation	6	Sound annotation.

4.10 CalendarFolderPublishTimePeriodConstants

These new constants can be returned from the folder objects PublishCalendarTimePeriod property.

Constant	Value	Description
egwPublishZeroDays	0x0000	Publish zero days before or after today.
egwPublishOneDay	0x0001	Publish items that fall within one day plus or minus of today.
egwPublishThreeDays	0x0002	Publish items that fall within three days plus or minus of today.
egwPublishSevenDays	0x0003	Publish items that fall with in seven days (1 week) plus or minus of today.
egwPublishFourteenDays	0x0004	Publish items that fall within fourteen days (2 weeks) plus or minus of today.
egwPublishThirtyDays	0x0005	Publish items that fall within thirty days (1 month) plus or minus of today.
egwPublishSixtyDays	0x0006	Publish items that fall within sixty days (2 months) plus or minus of today.
egwPublishNinetyDays	0x0006	Publish items that fall within ninety days (3 months) plus or minus of today.

4.11 CategoryParentObjectTypeConstants

The following constants allow users of the Categories collection object to determine what type of Parent object will be returned after calling the Categories.Parent() property:

Constant	Value
egwAddressBookEntryParent	0
egwAppointmentMessageParent	1
egwDocumentReferenceParent	2
egwMailMessageParent	3

Constant	Value
egwNoteMessageParent	4
egwPhoneMessageParent	5
egwTaskMessageParent	6
egwQuickMessageParent	7

4.12 CellPhoneCarrierConstants

These constants identify supported cell phone carriers.

--For GroupWise 2012 and later.

Constant	Value
egwMobileCarrierUnknown	0
egwMobileCarrierAlltel	1
egwMobileCarrierFirstLarge	1
egwMobileCarrierAtt	2
egwMobileCarrierBoostMobile	3
egwMobileCarrierNextel	4
egwMobileCarrierSprint	5
egwMobileCarrierTMobile	6
egwMobileCarrierUsCellular	7
egwMobileCarrierVerizon	8
egwMobileCarrierVirginMobile	9
egwMobileCarrierLastLarge	9
egwMobileCarrier711	10
egwMobileCarrierFirstSmall	10
egwMobileCarrierAirTel	11
egwMobileCarrierAireTelWireless	12
egwMobileCarrierAlaska	13
egwMobileCarrierAQL	14
egwMobileCarrierATTPaging	15
egwMobileCarrierBigRedGiant	16
egwMobileCarrierBellMobility	17
egwMobileCarrierBPL	18
egwMobileCarrierCellularOne	19

Constant	Value
egwMobileCarrierCingularPost	20
egwMobileCarrierCentennial	21
egwMobileCarrierCingularPre	22
egwMobileCarrierClaroBZ	23
egwMobileCarrierClaroN	24
egwMobileCarrierComcel	25
egwMobileCarrierCricket	26
egwMobileCarrierCTI	27
egwMobileCarrierEmTel	28
egwMobileCarrierFido	29
egwMobileCarrierGeneral	30
egwMobileCarrierGlobalStar	31
egwMobileCarrierHelio	32
egwMobileCarrierIllinois	33
egwMobileCarrierIridium	34
egwMobileCarrierIUSACell	35
egwMobileCarrierIWireless	36
egwMobileCarrierKoodo	37
egwMobileCarrierLMT	38
egwMobileCarrierMeteor	39
egwMobileCarrierMero	40
egwMobileCarrierMetropcs	41
egwMobileCarrierMovicom	42
egwMobileCarrierMobitel	43
egwMobileCarrierMovistar	44
egwMobileCarrierMTN	45
egwMobileCarrierMTS	46
egwMobileCarrierNextelUS	47
egwMobileCarrierNextelAR	48
egwMobileCarrierOrange	49
egwMobileCarrierPersonal	50
egwMobileCarrierPlus	51
egwMobileCarrierPresidents	52

Constant	Value
egwMobileCarrierQwest	53
egwMobileCarrierRogers	54
egwMobileCarrierSLI	55
egwMobileCarrierSaskTel	56
egwMobileCarrierSetar	57
egwMobileCarrierSunCom	58
egwMobileCarrierTMobileAU	59
egwMobileCarrierTMobileUK	60
egwMobileCarrierTelus	61
egwMobileCarrierTelCel	62
egwMobileCarrierThumb	63
egwMobileCarrierTigo	64
egwMobileCarrierTracFone	65
egwMobileCarrierUnicel	66
egwMobileCarrierViginMobileCA	67
egwMobileCarrierVodacom	68
egwMobileCarrierVodafone	69
egwMobileCarrierYCC	70
egwMobileCarrierMobiPCS	71
egwMobileCarrierVodafoneES	72
egwMobileCarrierVodafoneDE	73
egwMobileCarrierVodafoneUK	74
egwMobileCarrierVodafoneNZ	75
egwMobileCarrierTMobileHR	76
egwMobileCarrierTMobileDE	77
egwMobileCarrierTMobileNL	78
egwMobileCarrierTMobileAT	79
egwMobileCarrierEPlus	80
egwMobileCarrierO2	81
egwMobileCarrierO2UK	82
egwMobileCarrierO2DE	83
egwMobileCarrierLastSmall	83
egwMobileCarrierCustom	1000

4.13 CheckInConstants

These constants identify the type of action performed when checking-in a document version.

Constant	Value	Description
egwDoCheckIn	0	Change version status to Checked In.
egwLeaveCheckedOut	1	Copy in a new version, and leave this version checked out.

4.14 ChecklistSequenceConstants

The constants identify the checklist sequence:

Constant	Value	Description
egwChecklistTop	0	Item moves to the top of the checklist.
egwChecklistBottom	1	Item moves to the bottom of the checklist.
egwChecklistBetween	2	Item moves between two other messages on the checklist.

4.15 DisposalMethodConstants

These constants identify the methods of disposing of a document.

Constant	Value	Description
egwDisposeByArchive	1	Dispose of the document by archiving it.
egwDisposeByDelete	2	Dispose of the document by deleting it.

4.16 DocumentAccessRightsConstants

These constants identify the type of document access rights granted to individual users or groups.

Constant	Value	Description
egwARAllowDelete	1	The user can delete the specified document versions.
egwARAllowEdit	2	The user can modify the specified document versions.
egwARAllowView	4	The user can view the specified document versions.
egwARModifySecurity	8	The user can modify the access rights of the owning document.
egwARRevokeAllRights	16	Revoke all user access rights. If set, the other bits are ignored.
egwARAllowShare	32	The user can share viewing of the specified document version.

4.17 DocumentReferenceTypeConstants

These constants identify the version of a document reference.

Constant	Value	Description
egwOfficial	0	Refers to the official version, even if the official version changes.
egwCurrent	1	Refers to the most current version.
egwSpecific	2	Refers to a specific version.

4.18 DownloadStateConstants

These constants identify whether or not information has been downloaded.

Constant	Value	Description
egwAvailable	0	The information has been downloaded.
egwNotDownloaded	1	The information has not been downloaded.

4.19 EmailAddressFormatConstants

These constants identify the format of an email address.

Constant	Value	Description
egwEmailAddress4x	0	GroupWise 4.x.
egwEmailAddress5x	1	GroupWise 5.x.
egwEmailAddressTokenAPI	2	GroupWise 6.5 and later releases. If this constant is passed to the Account.ConvertEmailAddress method, an email address of UserID.PostOffice.Domain (UDP) is passed back. This UDP can be passed (through the Token API) to the GroupWise Windows client.

4.20 EndRetrieveStatus

These constants identify the type of action performed when ending the retrieval of a document version.

Constant	Value	Description
egwDoEndRetrieve	0	Mark the version as being available.
egwLeaveInUse	1	Leave the version marked as still in use.

4.21 EventTypeConstants

These constants identify the type of events that have been performed on a document version.

Constant	Value	Description
egwUnknownEvent	0	Unknown Event.
egwAdd	1	The document version was added.
egwCheckIn	2	The document version was checked in.
egwCheckInRequest	3	A request to check in the document version was uploaded.
egwCheckOut	4	The document version was checked out.
egwCheckOutRequest	5	A request to check out the document version was uploaded.
egwCopyIn	6	The document version was copied from disk to the library.
egwCopyOut	7	The document version was copied from the library to disk.
egwCopyVersion	8	A copy of the document version was made in the library.
egwDeleteVersion	9	The document version was deleted.
egwDeleteVersionRequest	10	A request to delete the document version was uploaded.
egwDownLoadRequest (L to l)	11	A request to download the document version was uploaded.
egwEndRetrieve	12	The retrieval of the document version was ended.
egwPreview	13	The document version was previewed.
egwRetrieve	14	The document version was retrieved.
egwSyncWithMasterRequest	15	A request, to synchronize the document version with the version in the master library, was uploaded.
egwUploadRequest	16	A request to upload the document version was uploaded.
egwCopyOutForPrint	17	The document version was copied out to disk for printing.
egwCopyOutForJefferson	18	The document version was copied out to disk for GroupWise Web Publisher.
egwSecurityChange	19	The document version’s security was changed.
egwDocumentShared	20	The document version was shared with other users.
egwMakeVersionOfficial	21	The document version was made the official version.
egwUpdateFields	22	The document version’s fields were updated.
egwDocUpAddServer	23	A new document version was uploaded from Remote.
egwDocUpModServer	24	A modified document version was uploaded from Remote.
egwRemoteResetInMaster	25	The document version’s status was reset by a Remote system.
egwVersionDownClientRemote	26	The document version was downloaded to Remote.
egwDocUpClientRemote	27	The document version was uploaded from Remote.
egwRemoteReset	28	The document version’s status was reset on a Remote system.
egwMove	29	The document version was moved to a new library.
egwStatusReset	30	The document version’s status was reset.
egwArchived	31	The document version was archived.

Constant	Value	Description
egwRestored	32	The document version was restored.
egwBackedUp	33	The document version was backed up.
egwRestoredBackUp	34	The document version was restored from a backup.
egwRemoveSyncWithMasterRequest	35	A request to remove the synchronization of the document version with the version in the master library was uploaded.
egwDocumentAvailableOnServer	36	The document version was made available on the server.
egwDownloadFailed	37	Downloading the document version failed.
egwVersionDeleteFailed	38	The document version was previously deleted from the server.

4.22 FieldDefinitionCase

These constants identify the string case for a field definition.

Constant	Value	Description
egwUpperCase	1	Upper case.
egwLowerCase	2	Lower case.
egwMixedCase	3	Mixed case.

4.23 FieldTypeConstants

These constants identify the type of a user-defined field.

Constant	Value	Description
egwString	1	String.
egwNumeric	2	Numeric (32 bits).
egwDate	3	Date.
egwBinary	4	Binary.
egwReserved	5	Reserved.

4.24 FolderRightsConstants

These constants identify the type of access rights a user has to a folder.

Constant	Value	Description
egwAllowDelete	1	The user can delete items from the folder.
egwAllowAdd	2	The user can add items to the folder.
egwAllowModify	4	The user can modify items in the folder.

4.25 FolderRightsNotifyConstants

These constants identify the group of users being notified of a change in their access rights to a shared folder.

Constant	Value	Description
egwDeleted	1	Notify users who have had their access rights to the shared folder revoked.
egwModified	2	Notify users who have had their access rights to the shared folder changed.
egwNew	3	Notify users who have just been given access rights to the shared folder.

4.26 FolderTypeConstants

The following constants identify a folder's type.

Constant	Value	Description
egwUnknownFolder	0	Unknown.
egwMailbox	1	Mailbox.
egwRoot	2	Root.
egwQuery	3	Query.
egwPersonalFolder	4	Personal.
egwCabinet	5	Cabinet.
egwCalendar	6	Calendar.
egwWork	7	Work In Progress.
egwDocuments	8	Documents.
egwIMap	9	IMAP account
egwNNTPNewsGroup	10	NNTP News
egwNNTPServer	11	NNTP Server
egwSentItems	12	Sent Items system
egwChecklist	13	Checklist
egwContacts	14	Contacts
egwSubCalendar	15	Folder contains calendar items. This folder exists under the main calendar folder.
egwUserContacts	16	Folder contains AddressBookEntries (contacts) from a users personal address book.
egwCalendarSubscribe	17	Folder contains Calendar items read from a subscribe to iCal web site.
egwRSS	18	Folder contains items pulled from a RSS feed.

4.27 FolderUpdateFrequencyConstants

These constants can be returned from the folder objects SubscribeCalendarFrequamcy property:

Constant Name	Value	Description
egwFolderUpdateManually	0x0000	Calendar items to be read only when manually requested.
egwFolderUpdateFifteenMinutes	0x0001	Calendar items to be read every 15 minutes.
egwFolderUpdateHourly	0x0002	Calendar items to be read every hour
egwFolderUpdateDaily	0x0003	Calendar items to be read daily.
egwFolderUpdateWeekly	0x0004	Calendar items to be read weekly.
egwFolderUpdateMonthly	0x0005	Calendar items to be read monthly.

4.28 IMAddressTypeConstants

These constants identify instant messaging programs:

Value	Constant Name	Description
0	egwIMAddressTypeUnknown	Unknown IM Address type
1	egwIMAddressTypeAIM	AIM IM Address type
2	egwIMAddressTypeMSN	MSN IM Address type
3	egwIMAddressTypeICQ	ICQ IM Address type
4	egwIMAddressTypeNovell	Novell IM Address type

4.29 LoginConstants

These constants control the login prompting if there is a problem logging in.

Constant	Value	Description
egwPromptIfNeeded	0	Prompt the user for the needed login information.
egwNeverPrompt	1	Never prompt the user. Throw an exception.
egwAllowPasswordPrompt	2	Prompt the user for a password.

4.30 MessageBoxTypeConstants

These constants identify the box type of a message.

Constant	Value	Description
egwIncoming	1	Incoming.
egwOutgoing	2	Outgoing.

Constant	Value	Description
egwPersonal	3	Personal.
egwDraft	4	Draft.

4.31 MessagePriorityConstants

These constants identify the priority of a message.

Constant	Value	Description
egwLow	1	Low.
egwNormal	2	Normal.
egwHigh	3	High.

4.32 MessagePropertiesConstants

These constants can be passed to the MessageProperty method of Appointment, Mail, Phone, Note, Task, Shared Notification, and Document References.

Constant	Value	Description
egwMessagePropertySenderUserID	0x0001	The Sender or User that created this message.
egwMessagePropertyRestoreDate	0x0002	If this message was restored from backup, the date and time of this restoration.
egwMessagePropertyUnarchiveDate	0x0003	If this message was unarchived, the date and time of this restoration.

4.33 MessageStatusConstants

This bitmask identifies the status of a message.

Constant	Value	Description
egwMessageStatusDefault	0	No status bits set for the message.
egwMessageOpened	1	Message opened.
egwMessageRead	2	Message read.
egwMessagePrivate	4	Message is private.
egwMessageReceiptRequested	16	Receipt notification requested.
egwMessageDeliverReceiptRequested	32	Delivery notification requested.

4.34 MessageSecurityConstants

These constants identify the security for a message.

Constant	Value	Description
egwDefaultSecurity	0	Default security.
egwNormalSecurity	1	Normal security.
egwProprietarySecurity	2	Proprietary.
egwConfidentialSecurity	3	Confidential.
egwSecretSecurity	4	Secret.
egwTopSecretSecurity	5	Top secret.
egwForYourEyesOnlySecurity	6	Your eyes only.

4.35 NotifyMessageConstants

These constants identify the type of notification for a message.

Constant	Value	Description
egwNoNotify	0	No notification needed.
egwSendReceipt	1	Send a notification message.
egwNotify	2	Notify the user of the message.
egwSendAndNotify	3	Send a notification message, and notify the user.

4.36 PhoneNumberConstants

These constants specify which phone number the given phone number is, as specified in the GroupWise contact.

--For GroupWise 2012 and later.

Constant	Value	Description
eGWPhoneUnknownPhone	0	The phone number type is unknown.
eGWPhonePrimaryPhone	100	The user’s primary phone number.
eGWPhoneAssistantPhone	101	The user’s assistant’s phone number.
eGWPhoneCallbackPhone	102	The user’s callback phone number.
eGWPhoneCarPhone	103	The user’s car phone number.
eGWPhoneCellPhone	104	The user’s cell phone number.
eGWPhoneFaxPhone	105	The user’s fax phone number.
eGWPhoneHomePhone	106	The user’s primary home phone number.

Constant	Value	Description
eGWPhoneHome2Phone	107	The user's secondary home phone number.
eGWPhoneISDNPhone	108	The user's Integrated Services Digital Network (ISDN) phone number.
eGWPhoneMainCompanyPhone	109	The user's company's phone number.
eGWPhoneOfficePhone	110	The user's primary office phone number.
eGWPhoneOffice2Phone	111	The user's secondary office phone number.
eGWPhoneOtherPhone	112	Another phone number for the user.
eGWPhonePagerPhone	113	The user's pager phone number.
eGWPhonePhone	114	Any other kind of phone number for the user.
GWPhoneRadioPhone	115	The user's radio phone number.
eGWPhoneSkypePhone	116	The user's Skype phone number.
eGWPhoneTelexPhone	117	The user's Telex phone number.

4.37 QuickMessagesCreationConstants

These constants allow users of the Account (page 26) and Folder (page 101) GetQuickMessagesCollection methods to determine what Message items are included in the collection.

Value	Constant
1	egwNewMessages
2	egwModifiedMessages
3	egwAllMessages

4.38 RecipientStatusConstants

These constants are returned from the StatusType property of the new RecipientStatus object.

Constant	Value	Description
egwRecipientAccepted	0x0001	Recipient accepted this calandar message.
egwRecipientCompleted	0x0002	Recipient completed this task.
egwRecipientDeclined	0x0003	Recipient declined this calandar message.
egwRecipientDelegated	0x0004	Recipient delegated this appointment to another user.
egwRecipientDeleted	0x0005	Recipient deleted this message.
egwRecipientDelivered	0x0006	This message was delivered to recipient.
egwRecipientDownloaded	0x0007	Recipient downloaded this message to a remote or caching database.

Constant	Value	Description
egwRecipientForwarded	0x0008	Recipient forwarded this message to other users.
egwRecipientIncomplete	0x0009	Recipient has not completed this task.
egwRecipientPending	0x000A	Transfer of this message to the recipient through the GWIA is pending.
egwRecipientPurged	0x000B	Recipient purged this message from their database.
egwRecipientRead	0x000C	Recipient read this message.
egwRecipientReplied	0x000D	Recipient replied to this message.
egwRecipientRetracted	0x000E	This message was retracted from the recipients mailbox.
egwRecipientRetractRequested	0x000F	Request to retract this message from recipient has been sent.
egwRecipientStarted	0x0010	Recipient started work on this message.
egwRecipientTransfered	0x0011	Message was transfered to the recipient through the GWIA.
egwRecipientTransferDelayed	0x0012	Transfer of the message to recipient has been delayed.
egwRecipientTransferFailed	0x0013	Transfer of the message to recipiant through the GWIA has failed..
egwRecipientThirdPartyDownloaded	0x0014	A third party product has downloaded this message into their store.
egwRecipientUndeleted	0x0015	Recipient undeleted this message.
egwRecipientUndeliverable	0x0016	Message is undeliverable to this recipient..
egwRecipientUnknownStatus	0x0017	Unknown status received from recipient account.

4.39 ReplyRequestedConstants

These constants identify when a reply is requested.

Constant	Value	Description
egwNoReply	0	No reply needed.
egwWhenConvenient	1	Reply when it is convenient.
egwWithinDaysRequested	2	Reply within the number of days specified by the ReplyDaysRequested property of the Message object.

4.40 ResolvedConstants

These constants identify the resolved status of a recipient.

Constant	Value	Description
egwNotResolved	0	The recipient is not resolved.

Constant	Value	Description
egwNotFound	1	The recipient could not be found in the address book.
egwAmbiguous	2	The recipient is ambiguous.
egwResolved	3	The recipient has been resolved.

4.41 RuleActionConstants

These constants identify the type of rule action to perform.

Constant	Value	Description
egwUnknownRuleAction	0	Unknown rule action.
egwAccept	1	Accept the item.
egwArchive	2	Archive the item.
egwCategory	3	Apply a category to the item.
egwDelegate	4	Delegate the item.
egwDelete	5	Delete the item.
egwForward	6	Forward the item.
egwLink	7	Link/Copy the item into a folder.
egwMarkPrivate	8	Mark the item private.
egwMarkRead	9	Mark the item read.
egwMarkUnread	10	Mark the item unread.
egwRuleMove	11	Move the item to a folder.
egwPurge	12	Delete and empty the item.
egwReply	13	Reply to the item.
egwReplyWithText	14	Reply with text to the item.
egwSend	15	Send an item.
egwStopRules	16	Stop processing rules on the item.

4.42 RuleConstants

These constants identify the type of rule event.

Constant	Value	Description
egwUnknownRule	0	Unknown rule event.
egwNewItem	1	New item delivered to the user.
egwStartup	2	When the GroupWise Client has completed starting up.

Constant	Value	Description
egwExit	3	When the GroupWise Client is about to shutdown.
egwFolderOpen	4	When the GroupWise Client selects a folder.
egwFolderClosed	5	When the GroupWise Client selects a different folder.
egwFiled	6	When an item is put into a folder.
egwCompleted	7	When an item is completed.
egwUserEvent	8	When the GroupWise Client manually executes a rule.

4.43 SharedFolderConstants

These constants identify a shared folder's type.

Constant	Value	Description
egwNotShared	0	This folder is not shared.
egwSharedOutgoing	1	This folder is shared with someone else.
egwSharedIncoming	2	Someone is sharing this folder with me.

4.44 SharedNotificationTypeConstants

These constants identify the types of shared notification messages.

Constant	Value	Description
egwSharedFolder	0	Shared folder.
egwSharedAddressBook	1	Shared personal address book.

4.45 SortConstants

These constants identify sort types.

Constant	Value	Description
egwAscending	0	Sorted in ascending order.
egwDescending	1	Sorted in descending order.

4.46 SynchronizeConstants

This bitmask indicates what is to be synchronized to a remote or client mailbox.

Constant	Value	Description
egwRequestItems	1	Requests items.
egwRequestNewItems	2	Requests new items.
egwRequestRules	4	Requests rules.
egwRequestSystemAddressBook	8	Requests the system address book.
egwRequestPersonalAddressBook	16	Requests the user's personal address books.
egwRequestProxy	32	Requests proxy information.
egwRequestLibraryList	64	Requests document library lists.
egwRequestAll	127	Requests all information.

4.47 TimeBlockConstants

These constants identify the type of a block of time.

Constant	Value	Description
egwFree	0	Free.
egwBlocked	1	Blocked.
egwOutOfOffice	2	Out of the office.
egwTentative	3	Tentative.

4.48 TimeZoneDaysOfWeekConstants

These constants are used when reading or writing the days of the week for the GroupWiseTimeZone object.

Constant	Value	Description
egwSunday	0x0001	Sunday
egwMonday	0x0002	Monday
egwTuesday	0x0003	Tuesday
egwWednesday	0x0004	Wenesday
egwThursday	0x0005	Thursday
egwFriday	0x0006	Friday
egwSaturday	0x0007	Saturday

4.49 TimeZoneMonthConstants

These constants are used when reading or writing the start and end Month for the GroupWiseTimeZone object.

Constant	Value	Description
egwJanuary	0x0001	January
egwFeburary	0x0002	Febuary
egwMarch	0x0003	March
egwApril	0x0004	April
egwMay	0x0005	May
egwJune	0x0006	June
egwJuly	0x0007	July
egwAugust	0x0008	August
egwSeptember	0x0009	September
egwOctober	0x000A	October
egwNovemberl	0x000B	November
egwDecember	0x000C	December

4.50 TimeZoneOccuranceConstants

These constants are used when reading or writing the start and end Occurances for the GroupWiseTimeZone object.

Constant	Value	Description
egwFirst	0x0001	First
egwSecond	0x0002	Second
egwThird	0x0003	Third
egwFourth	0x0004	Fourth
egwLast	0x0005	Last

4.51 VCardSourceConstants

These constrains can be returned or set from the Account object propertyVCardSource.

Constant	Value	Description
egwVCardUnknownSource	0x0001	VCard source has not been set.
egwVCardFileSource	0x0002	Attached VCard should be retreived from VCard Signature File.

Constant	Value	Description
egwVCardSABSource	0x0003	Attached VCard should be generated from the system address book entry of the user that is sendign this message.
egwVCardPABSource	0x0004	Attached VCard should be generated from the personal address book amd email address given in the Account property.

5 Filter Expressions

Object API methods that search for messages, address book entries, or documents in the user's database require a filter expression. Some examples are the Messages Object Find method and the Query Object property Expression.

This section covers the following topics:

5.1 Filter Expressions

A filter expression is a string. For a C++ developer, the Filter Expression is defined as type BString. For a Visual Basic or Delphi developer, the Filter Expression is a normal string. The String may contain one Simple Statement or multiple Simple Statements combined with Boolean Operators.

5.2 Simple Statements

A Simple Statement contains the following:

- An Opening Parenthesis character
- A Keyword
- An Operator
- A Value to use with the keyword
- A Closing Parenthesis

5.3 Keywords

Keywords represent a specific Property or Field in an item. For example, the keyword CREATE_DATE signals that this Filter will deal with the date an item was created.

For a list of available keywords, see

Keywords are separated by the following data types:

- Unary

- Date
- Numeric
- String
- Enumerated

Several system-defined keywords exist within each of the five data types. These are listed in item (e) below. (For example, the CREATE_DATE keyword above is a system-defined keyword of the DATE data type.) In addition, users can create their own custom defined keywords (equivalent to custom field definitions for messages, address books, or documents).

User Defined Fields can be used as Keywords by enclosing the name and type within "<>" characters. For example, if a GroupWise system has a User Defined field named "Bowling Shoe Size", a numeric field, it can be given as the "<Bowling Shoe Size, NUMERIC>" keyword.

- User Defined Field keywords must match the exact name given to the field.
- All Fields within an Address Book search are considered User Defined Fields of type STRING. The keyword syntax for address books can be shortened to <MyField> rather than <MyField, STRING>.

Keywords are not case sensitive. "CREATE_DATE" is the same as "Create_Date".

The following is a list of system-defined keywords, sorted by type.

Data Type	Keywords
Date	ASSIGNED_DATE CREATE_DATE DELIVERED_DATE DUEEND_DATE MODIFY_DATE RETRIEVED_DATE START_DATE
Enumerated	PRIORITY ATTACHMENT_TYPE BOX_TYPE
Numeric	CURRENT_VERSION_NUMBER DOCUMENT_NUMBER NUMBER_ACCEPTED NUMBER_COMPLETED NUMBER_DELETED NUMBER_OPENED NUMBER_READ NUMBER_REPLIED OFFICIAL_VERSION_NUMBER SIZE TOTAL_RECIPIENTS TASK_CATEGORY TASK_PRIORITY VERSION_NUMBER

Data Type	Keywords
String	ANY_FIELD AUTHOR ATTACHMENT_LIST BC CALLER_COMPANY CALLER_NAME CALLER_PHONE_NUMBER CATEGORY CC CLASS_NAME DOCUMENT_CREATOR DOCUMENT_FILENAME DOCUMENT_TYPE LIBRARY FROM MESSAGE PERSONAL_SUBJECT PLACE PRIMARY_CATEGORY RETRIEVED_BY SUBJECT TO VIEWNAME
Unary	ACCEPTED APPOINTMENT COMPLETED DELEGATED DELIVERED DOCREFERENCE DOCVERSION_ARCHIVED DOCVERSION_CHECKED_OUT DOCVERSION_IN_USE DOCVERSION_CONNECTED_READ_WRITE HIDDEN MAIL NOTE OFFICIAL_DOCUMENT_VERSIONS_ONLY ON_CALENDER ON_CHECKLIST OPENED PHONE_MESSAGE PRIVATE READ REPLY_REQUESTED ROUTED SEARCH_AS_LIBRARIAN TASK

5.3.1 Available Keywords

The following keywords are available:

Keyword	Description
ACCEPTED	UNARY keyword. Finds all TASK, NOTE, or APPOINTMENT items that were accepted by their recipient(s).
ANY_FIELD	STRING keyword. Searches most text fields for the given value.
APPOINTMENT	UNARY keyword. Finds all APPOINTMENT Message items.
ASSIGNED_DATE	DATE keyword. The date and time a TASK item was assigned to this user. (Note: a task can be assigned by another person or by creating a Personal Task.)
ATTACHMENT_LIST	STRING keyword. Obsolete. Use the ATTACHMENT_TYPE keyword instead.
ATTACHMENT_TYPE	ENUMERATED keyword. The type of item attached to any MESSAGE. (APPOINTMENT, MAIL, NOTE, TASK, PHONE MESSAGE, DOCUMENT REFERENCE, or SHARED NOTIFICATION items.) The value to search for must come from this list of text constants: FILE, MESSAGE, SOUND, APPOINTMENT, TASK, NOTE, MAIL, MOVIE, PHONE_MESSAGE, OLE, or DOCREFERENCE.
AUTHOR	STRING keyword. Searches a LIBRARY for any DOCUMENT written by a given user.
BC	STRING keyword. Searches MESSAGE items for a Blind Copy recipient.
BOX_TYPE	ENUMERATED keyword. The type of any MESSAGE item in the users mailbox. The constant value INCOMING signifies a group item that was delivered to this user. The value OUTGOING identifies a message sent to others. The value PERSONAL defines an item that was created in and exists only in this users mailbox. Finally, the value DRAFT identifies a MESSAGE item that is in progress, and may become an OUTGOING item at a later date.
CALLER_COMPANY	STRING keyword. Searches a PHONE MESSAGE item for the company name of the caller. Searches an ADDRESS BOOK ENTRIES collection for the company name of each Contact.
CALLER_NAME	STRING keyword. Searches a PHONE MESSAGE item for the name of the caller.
CALLER_PHONE_ NUMBER	STRING keyword. Searches a PHONE MESSAGE item for the phone number of the caller.
CATEGORY	STRING keyword. Searches a MESSAGE item for a CATEGORY. The CATEGORY text value must be defined in the CATEGORY DEFINITIONS collection. Even though the CATEGORY keyword is a string, it can be used only with the Matches operator. It cannot be used with the Contains, BeginsWith, or DoesNotContain operators.
CC	STRING keyword. Searches a MESSAGE item for a recipient name inside the Carbon Copy list.

Keyword	Description
CLASS_NAME	STRING keyword. Searches a MESSAGE item for a specific sub-type of message. The Class Name for normal mail messages is "GW.MESSAGE.MAIL". A sub-type of message may add to that Class Name string more text. For example, "GW.MESSAGE.MAIL.CRITICAL" would add the sub-type "CRITICAL" to the MESSAGE item.
COMPLETED	UNARY keyword. Returns all TASK items that have been marked Completed.
CREATE_DATE	DATE keyword. The date and time an item was created in the users mailbox.
CURRENT_VERSION_NUMBER	NUMERIC keyword. The latest version number for a DOCUMENT in a LIBRARY.
DELEGATED	UNARY keyword. Returns all MESSAGE items that were Delegated to another user by the original recipient(s).
DELIVERED	UNARY keyword. Searches for all MESSAGE items that were Delivered to their recipient(s).
DELIVERED_DATE	DATE keyword. The date and time a group item was delivered to the users mailbox.
DOCUMENT_CREATOR	STRING keyword. Searches a DOCUMENT for the name of the user that created this document.
DOCUMENT_FILENAME	STRING keyword. Searches a DOCUMENT item for the specific file name that this document is known by in the Operating System. This file name will only be present if this document has been checked out or saved to the local computer.
DOCUMENT_NUMBER	NUMERIC keyword. A number assigned to a DOCUMENT in a LIBRARY.
DOCUMENT_TYPE	STRING keyword. Searches a DOCUMENT item for a DOCUMENT TYPE. The DOCUMENT TYPE text must be defined as part of the DOCUMENT TYPES collection.
DOCREFERENCE	UNARY keyword. Returns all MESSAGE items that are references to DOCUMENT objects.
DOCVERSION_ARCHIVED	UNARY keyword. Searches for all DOCUMENT items that have a Version archived.
DOCVERSION_CHECKED_OUT	UNARY keyword. Searches for any DOCUMENT items that have a status of Checked Out.
DOCVERSION_IN_USE	UNARY keyword. Searches for any DOCUMENT items that have a status of In Use.
DOCVERSION_CONNECTED_READ_WRITE	UNARY keyword. Returns any DOCUMENT item that are marked as Connected Read Write.
DUEEND_DATE	DATE keyword. The date and time a TASK item must be completed by. The date and time an APPOINTMENT or NOTE item ends.
FROM	STRING keyword. Searches a MESSAGE item for a recipient name in the FROM field.
HIDDEN	UNARY keyword. Returns any MESSAGE item that has been marked as hidden.

Keyword	Description
LIBRARY	STRING keyword. Searches for DOCUMENT items that belong to a specific LIBRARY.
MAIL	UNARY keyword. Returns any MESSAGE item of type Mail.
MESSAGE	STRING keyword. Searches the Message Body of a MESSAGE item for the given text.
MODIFY_DATE	DATE keyword. The date and time an item in the user's mailbox was last modified.
NOTE	STRING keyword. Returns any MESSAGE item of type NOTE (a.k.a. Reminder Note).
NUMBER_ACCEPTED	NUMERIC keyword. The number of users who have accepted a TASK, NOTE, or APPOINTMENT item
NUMBER_COMPLETED	NUMERIC keyword. The number of users who have completed a TASK.
NUMBER_DELETED	NUMERIC keyword. The number of users who have deleted a MESSAGE.
NUMBER_OPENDED	NUMERIC keyword. The number of users who have opened a MESSAGE.
NUMBER_READ	NUMERIC keyword. The number of users who have read a MESSAGE.
NUMBER_REPLIED	NUMERIC keyword. The number of users who have replied to a MESSAGE.
OFFICIAL_DOCUMENT_VERSIONS_ONLY	UNARY keyword. While searching LIBRARIES for a DOCUMENT, return only Document References to the "official" version of this document.
OFFICIAL_VERSION_NUMBER	NUMERIC keyword. The version number of a DOCUMENT that is considered to be the "official" version.
ON_CALENDAR	UNARY keyword. Searches for any APPOINTMENT, NOTE, or TASK item. (FYI: Only Appointments, Notes, or Tasks are considered to be "On the calendar".)
ON_CHECKLIST	UNARY keyword. Returns any MESSAGE item that has been placed on the "Checklist".
OPENED	UNARY keyword. Searches for all MESSAGE items that have been opened by the recipient. Note: Personal or Draft items are considered "opened" when they are created.
PERSONAL_SUBJECT	STRING keyword. Searches any MESSAGE item for the Subject assigned by the recipient.
PHONE_MESSAGE	UNARY keyword. Returns all MESSAGE items of type Phone Message.
PLACE	STRING keyword. Searches an APPOINTMENT item for the location of this appointment.
PRIMARY_CATEGORY	STRING keyword. Searches any MESSAGE item for the CATEGORY assigned as the Primary Category. The CATEGORY text value must come from the CATEGORIES collection.
PRIORITY	ENUMERATED keyword. The Priority placed on a message item. Values to search for must be on of these text strings: HIGH, LOW, or NORMAL.
PRIVATE	UNARY keyword. Returns all MESSAGE items that are marked as Private.

Keyword	Description
READ	UNARY keyword. Returns all MESSAGE items that have been Read by the recipient. Note: Draft and Personal items are considered to have been Read at all times.
REPLY_REQUESTED	UNARY keyword. Searches for all MESSAGE items that the Sender asked for all Recipients to Reply to.
RETRIEVED_BY	STRING keyword. Searches DOCUMENT items for the name of the last person to Check Out or Save the document.
RETRIEVED_DATE	DATE keyword. The most recent date and time a DOCUMENT item was retrieved from a library.
ROUTED	UNARY keyword. Returns any MESSAGE item that are marked as part of a Work Flow.
SEARCH_AS_ LIBRARIAN	UNARY keyword. Users may be granted LIBRARIAN rights to any DOCUMENT LIBRARY. Setting this keyword allows the current user to use those rights to search all DOCUMENTS in the LIBRARY. Without this keyword, the returned list of DOCUMENT REFERENCES will only contain DOCUMENTS that the user may see using normal LIBRARY security.
SIZE	NUMERIC keyword. The size of an item in the mailbox.
START_DATE	DATE keyword. The date and time a TASK, REMINDER NOTE, or APPOINTMENT item begins.
SUBJECT	STRING keyword. Searches a MESSAGE items Subject field.
TASK	UNARY keyword. Returns all MESSAGE items of type TASK.
TASK_CATEGORY	NUMERIC keyword. This is listed as a NUMERIC keyword, and uses the NUMERIC data operators for all searches. But all values to search for must be a single text character from A-Z. For example, a TASK may be assigned a Category of "A" and a Priority of "4". The GroupWise Windows client will display this as the string "A4" in the Priority window. To find all TASKS assigned with a Category of "A", construct a filter expression similar to this: "(TASK_CATEGORY = "A")".
TASK_PRIORITY	NUMERIC keyword. The numeric priority given to a TASK. For example, a TASK item may be given a priority of "B" and a Priority of 2. If this TASK is displayed in the GroupWise Windows client, the Priority window will display the value "B2".
TO	STRING keyword. Searches a MESSAGE item for a recipient in the TO field.
TOTAL_RECIPIENTS	NUMERIC keyword. The number of users that this MESSAGE was sent to.
VERSION_NUMBER	NUMERIC keyword. A revision number of a DOCUMENT within a LIBRARY.
VIEW	STRING keyword. Obsolete. Searches a MESSAGE item for a name that displays a specific View in the GroupWise Windows Client. (Remember the old View Designer? The name used inside that product was stored in each MESSAGE item that should be displayed using that form.)

5.3.2 AddressBook Keywords

The keywords for the GroupWise System Address Book are

- Name

- First Name
- Last Name
- E-Mail Address
- Department

For example, you can use filters similar to the following:

(Name CONTAINS "Smi*")
(<First Name> MATCHES "John")
(<Last Name> CONTAINS "Smi*")
(<E-Mail Address> CONTAINS "*novell.com")
(Department CONTAINS "System Test")

NOTE: Field names that contain spaces must be wrapped between brackets (< >).

5.4 Operators

Each data type has specific operators that can be used with a given keyword.

This section covers the following topics:

5.4.1 UNARY

UNARY keywords have no operators (or Value data). Follow a UNARY keyword filter expression with either a Closing Parenthesis or another Simple Statement. For example, the filter expression "(APPOINTMENT)" will find all Appointment messages.

There is one exception to this rule. The NOT operator can proceed a UNARY keyword. The following are a few examples:

- To find all messages that are not appointments, use (NOT APPOINTMENT).
- If you want to find the item status of messages that are not read, use (NOT READ).
- If you want to find the item status of messages that are not opened, use (NOT OPENED).

5.4.2 Date

Date keywords use the following operators:

Operator	Description
=	Equal to
>=	Greater than or equal to

Operator	Description
>	Greater than
<=	Less than or equal to
<	Less than

5.4.3 Numeric

Numeric keywords use the following operators:

Operator	Description
=	Equal to
>=	Greater than or equal to
>	Greater than
<=	Less than or equal to
<	Less than
<>	Not equal to

5.4.4 String

String keywords use the following operators:

Operator	Description
Matches	Equals, exactly matches the string Value
Contains	Field has the value string in it
BeginsWith	Field has the value string as the first characters of the data string
DoesNotContain	Field does not contain the value string

NOTE: String operators are not case sensitive.

5.4.5 Enumerated

Enumerated keywords use the following operators:

Operator	Description
=	Equal to
<>	Not equal to

5.5 Text Strings

Each type of keyword requires a text string representing a value to use in the search. This value must be formatted according to the keyword data type used.

This section covers the following topics:

5.5.1 UNARY Strings

UNARY keywords have no Operator or Values. A UNARY keyword is immediately followed by the close of a simple statement or by a BOOLEAN operator. For example, if you want to search for all hidden messages inside a folder, you would construct the following filter expression: "(HIDDEN)".

5.5.2 Date Strings

DATE keywords will be followed by a DATE operator, then with a Date Value. The Date Value consists of one of the following:

Data Value	Description
Date Constant	A Date Constant must be typed in exactly as given in the strings. Date Constants are not case sensitive. The following is a list of date constants: ◆ TODAY ◆ YESTERDAY ◆ TOMORROW ◆ THIS_WEEK ◆ THIS_MONTH ◆ THIS_YEAR A Date Constant may be followed by a Relative Date Value. A Relative Date Value is a numeric text string, such as "99". This Date Value will be interpreted as a number to be added to the Date Constant. If the "+" or "-" sign is missing from the number, it will default to addition. For example, if the filter is looking for a date that must be less than 7 days in the future, the data value would be "TODAY 7" or "TODAY + 7". **NOTE:** The numeric will be translated relative to the time period of the Date Constant. If the Date Constant refers to a day, the numeric value will be translated to days; if the Date Constant refers to a month, the numeric value will be the number of months.

Data Value	Description
Text string representing a specific date	A text string representing a specific date, given as Year/Month/Day. If an exact time within a given date is needed, the text string will be Year/Month/Day AT Hour:Minute:Second. For example, to look for the beginning of the year 2012, the Date Value will be "2012/01/01". If the filter needed to look for events happening at Noon on the first day of the year 2012, the Date Value would be "2012/01/01 AT 12:0:0". The Year, Month, and Day must be given as a Number - no Names of the Month or Day are allowed. For example, "January" is an invalid Month value; use "01" instead.
Text string representing a specific time	A text string representing a specific time, in the format of Hour:Minute:Second. Note: The Minute and Second values are optional, and may be dropped. The Hour value is based on a 24 hour clock. The Time value will be compared based on the time zone used on the computer. Here are a few examples of valid Time values: ◆ 12:00:00 (Noon) ◆ 13 (1:00 p.m.) ◆ 8:30:30 (30 seconds past 8:30 a.m.)
Another keyordw of type Date	Another Keyword of type Date. For example, the filter expression "CREATE_DATE = MODIFIED_DATE" will find all items that have not been modified since they were created.

5.5.3 Numeric Strings

NUMERIC keywords may be either a valid number or another NUMERIC keyword.

If a text string is given, it must have a valid number as the Value. These numbers must be valid Double Word Integers (i.e. less than 4,294,967,295), with no decimal places or floating point representation allowed. For example, "99".

Another NUMERIC keyword may also be given as the Data Value. If searching for all Document Reference items where the Current Version Number equals the Official Version Number, the filter expression should read "(CURRENT_VERSION_NUMBER = OFFICIAL_VERSION_NUMBER)"The Numeric keyword TASK_CATEGORY is an exception to this rule. When a Task item is created, it is given a single alphabetic character value from A-Z and a numeric value (1-999). This is similar to the system used in popular paper Day Planners. For example, my highest priority task would be given as "A1". When searching, the data operator comes from the Numeric keyword operators, but the Value must be given as a single character string. For example, if I wanted to find all Tasks that had Task Category "A", the simple filter expression would be (TASK_CATEGORY = "A").

5.5.4 String Keywords

STRING keywords require a text value or another STRING keyword.

If a text string is given, the value must be placed between Double Quote marks. For example, "Now hear this" would search inside the text field for the string ’Now hear this’. Note: string values are not case sensitive.

To place a Double Quote character inside the Value, use two Double Quote marks in a row inside the text Value. For example, (SUBJECT CONTAINS "Find a "" Double Quote Character") would search for the text 'Find a " Double Quote Character' inside the Subject field.

The text operators CONTAINS and DOESNOTCONTAIN have a the following special features:

- The use of the wildcard characters "*" and "?" is allowed. When the Asterisk character is included in the text string, the text pattern can be followed by any other set of characters to be deemed a match. For example, the string "Attach*" would match on the words "Attach" and "Attachment". The Question Mark character signifies that any character may appear in this spot of the text pattern. For example, the text constant "D?d" would find the words "Did" and "Dad inside the search field.
- The character string ".." can be part of the text value. The convention ".." means "followed by." For example, the filter expression (SUBJECT CONTAINS "Test" .. "Plan") means find the words "Test" and "Plan" in that order, with any number of characters between them.
- The Boolean operators "AND" and "OR" may be used between text constants. For example, the filter expression "MESSAGE CONTAINS ("Internet" OR "Intranet")" searches for all messages that contain the words Internet or Intranet in their message body.
- The text constant may be proceeded with the following keywords. The keywords "NOCASE", "NOPREFIX", and "WILDCARD" are turned on by default.

  CASE (case sensitive search)

  NOCASE (case insensitive search)

  NOPREFIX (words are not automatically treated as wildcard prefix strings)

  NOSCAN (do not scan search)

  NOWILDCARD (wildcards are not expanded)

  PREFIX (Words are treated as if they were the prefix to a wildcard search. For example the text "Test Plan" would change to be the words "Test* Plan*". This is not valid if the NOWILDCARD constant is used.)

  RECORD (search results are found within Mail Merge document records)

  SUMMARY = n (Word Perfect Document Summaries are searched)

  WILDCARD (use the characters "*" and "?" as wildcards).

Another STRING keyword may be given as the data value. In this case, the GroupWise system will compare the text inside both STRING keyword fields to determine if the item matches the Filter Expression. For example, I could search for a message that contains the User Defined field called "Book Subject" inside the SUBJECT of the message, I would construct the Filter Expression "(SUBJECT CONTAINS <Book Subject, STRING>)"

5.5.5 Enumerated Strings

ENUMERATED values must come from the list that belongs to each Enumerated keyword.

Keyword Value	Values
PRIORITY	Low High Normal

Keyword Value	Values
ATTACHMENT_TYPE	FILE MESSAGE SOUND APPOINTMENT TASK NOTE MAIL MOVIE PHONE_MESSAGE OLE DOCREFERENC
BOX_TYPE	INCOMING OUTGOING PERSONAL DRAFT

5.6 Compound Statements

A compound statement consists of two or more simple statements that have been connected by the Boolean operators "AND" or "OR". The completed statement must have matching opening and closing parenthesis characters.

The Boolean "AND" or "OR" operators may be placed inside a set of matching parenthesis. For example - "(SUBJECT CONTAINS "TEST" OR SUBJECT CONTAINS "PLAN")" will find any message with the words "test" or "plan" in the subject.

The Boolean operators may also be placed between simple statements. For example: "(AUTHOR CONTAINS "JOE") AND (AUTHOR CONTAINS "BOB")" will find any document that was written by someone with the name "Joe" or "Bob".

The operators may appear in both places, as outlined in sections a) and b) (above). An illustration is better than a 1000 words here: "(APPOINTMENT AND START_DATE = TODAY) OR (NOTE AND START_DATE = TODAY) OR (TASK AND START_DATE = TODAY)" will find any Appointment, Reminder Note, or Task that belongs to the current day.

5.7 Examples of Valid Filter Expressions

This section contains samples of valid filter expressions.

For the address book, you can use filters similar to the following:

(Name CONTAINS "Smi*")
(<First Name> MATCHES "John")
(<Last Name> CONTAINS "Smi*")
(<E-Mail Address> CONTAINS "*novell.com")
(Department CONTAINS "System Test")

NOTE: Field names that contain spaces must be wrapped between brackets (< >).

The following filter expressions work for other queries:

```
(SUBJECT CONTAINS "Internet")
(AUTHOR MATCHES "Ralph Spoilsport")
(MESSAGE BEGINSWITH "Now Hear This")
(MESSAGE CONTAINS CASE ('Inter*' OR 'Intra*') AND ("Test" .. "Plan"))
(FROM MATCHES <MY_BOSS, STRING>)
(TASK)
(CREATE_DATE >= YESTERDAY AND CREATE_DATE <= TODAY)
(FROM CONTAINS "TJEFFERSON" OR FROM CONTAINS "JADAMS") AND (MESSAGE
   BEGINSWITH "When in the course" AND CREATE_DATE = 1776/7/4)
(DUEEND_DATE <= TOMORROW)
(START_DATE >= 2012/2/5 AT 8:00:00)
(CREATE_DATE >= THIS_YEAR 31)
(<BIRTHDAY, DATE> = THIS_MONTH)
(SIZE < 12000)
(NUMBER_ACCEPTED = TOTAL_RECIPIENTS)
(<TOTAL_EMPLOYEES, NUMERIC> > 50)
(PRIORITY = HIGH)
(ATTACHMENT_TYPE = OLE)
(BOX_TYPE = INCOMING)
(APPOINTMENT AND NOT ACCEPTED)
(MAIL OR TASK) AND (NOT HIDDEN)
```

6 Sample Applications

The following GroupWise samples are available for download from the GroupWise Object API page (http://developer.novell.com/wiki/index.php/GroupWise_Object_API) on the Novell Developer Kit (NDK):

For more information about the samples that are described in this section, see the GroupWise Object API Sample Code page (../../../samplecode/gwobjapi_sample/index.htm) on the NDK.

6.1 ABFind

Allows you to find Address Book entries using AddressBookEntries.Find("Filter Expression").

This sample can be downloaded from the GroupWise Object API page (http://developer.novell.com/wiki/index.php/GroupWise_Object_API) on the NDK and viewed at ABFIND.FRM (../../../samplecode/gwobjapi_sample/ABFIND/VB/ABFIND.FRM.html).

6.1.1 Filter Expression Example

```
(<First Name> MATCHES "Dee")
```

Make sure you include the parentheses.

6.1.2 Object API Objects

This sample uses the following Object API objects:

Account
AddressBook
AddressBookEntries
AddressBookEntry
Addresses

6.2 Address

Displays all the AddressBooks on a particular post office, All AddressBookEntries for a selected AddressBook, and AddressBookEntry information for a selected AddressBookEntry.

This sample can be downloaded from the GroupWise Object API (http://developer.novell.com/wiki/index.php/GroupWise_Object_API) page on the NDK and viewed at ADDRESS.FRM (../../../samplecode/gwobjapi_sample/ADDRESS/VB/ADDRESS.FRM.html).

6.2.1 Object API Objects

This application uses the following Object API objects:

Account
Address
AddressBook
AddressBookEntries
AddressBooks
Addresses

6.3 Appt

Demonstrates how to create an appointment using the Object API.

This sample can be downloaded from the GroupWise Object API (http://developer.novell.com/wiki/index.php/GroupWise_Object_API) page on the NDK and viewed at APPT.CPP (../../../samplecode/gwobjapi_sample/APPT/CPP/APPT.CPP.html).

6.3.1 Object API Objects

This application uses the following Object API objects:

Account
Address
Appointment
BusySearchElements
BusySearchResult
Folder
Message
Messages
Recipient
Recipients
TimeBlock
TimeBlocks

6.4 Login

Shows how to log in to the Object API., use the Object API to get to the root folder, look at the folders in the root folder, read messages in from a message collection, and create a new message.

This sample can be downloaded from the GroupWise Object API (http://developer.novell.com/wiki/index.php/GroupWise_Object_API) page on the NDK and viewed at BS1.FRM (../../../samplecode/gwobjapi_sample/LOGIN/VB/BS1.FRM.html).

6.4.1 Object API Objects

This application uses the following Object API objects:

Account
Application
Folder
Folders
Message
Messages

6.5 Mail

Demonstrates how to create and send mail items using the Object API.

This sample can be downloaded from the GroupWise Object API (http://developer.novell.com/wiki/index.php/GroupWise_Object_API) page on the NDK and viewed at SENDMAIL.CPP (../../../samplecode/gwobjapi_sample/MAIL/CPP/SENDMAIL.CPP.html).

6.5.1 Object API Objects

This application uses the following Object API objects:

Account
AddressBook
AddressBookEntries
AddressBookEntry
Application
Attachment
Attachments
Folder
Folders
FormattedText
Message
Messages
Recipient
Recipients
Trash
TrashEntries
TrashEntry

6.6 MsgFind

Allows you to find messages using Messages.Find("Filter Expression").

This sample can be downloaded from the GroupWise Object API (http://developer.novell.com/wiki/index.php/GroupWise_Object_API) page on the NDK and viewed at MSGFIND.FRM (../../../samplecode/gwobjapi_sample/MSGFIND/VB/MSGFIND.FRM.html).

6.6.1 Filter Expression Example

```
(SUBJECT CONTAINS "IMPORTANT:").
```

Note: Make sure you include the parentheses.

6.6.2 Object API Objects

This application uses the following Object API objects:

Account
Application
Folders
Message
MessageList
Messages

6.7 NCC

This sample application demonstrates how the GroupWise AddressBook and Name Completion OCX controls operate. It displays and allows you to add information to the Name Completion control. It also displays information from the AddressBook control.

This sample can be downloaded from the GroupWise Object API (http://developer.novell.com/wiki/index.php/GroupWise_Object_API) page on the NDK and viewed at GWNCC.FRM (../../../samplecode/gwobjapi_sample/NCC/VB/GWNCC.FRM.html).

6.8 POP3

Uses a modified version of the Delphi 2.01 POP3 mail example to add POP3 mail messages to your GroupWise mailbox.

This sample can be downloaded from the GroupWise Object API (http://developer.novell.com/wiki/index.php/GroupWise_Object_API) page on the NDK and viewed at MAIN.PAS (../../../samplecode/gwobjapi_sample/POP3/DELPHI20/MAIN.PAS.html).

6.8.1 Object API Objects

This application uses the following Object API objects:

Account
Application
Message

Messages
Recipient
Recipients

6.8.2 Remarks

POP3 requires the following: NetManage OCX and the DLLs that are included with Delphi 2.01 (NMOCOD.DLL, NMORENU.DLL, NMSCKN.DLL, and POPCT.OCX).

Make sure NMOCOD.DLL and the POPCT.OCX file are registered on the machine this sample application is used on.

The POP3 interface is very basic. It does not delete any mail from the POP3 server nor does it check for duplicate messages that may have been added to your GroupWise mailbox from prior sessions.

6.9 Query

Shows how to create a message store query using the Object API.

This sample can be downloaded from the GroupWise Object API (http://developer.novell.com/wiki/index.php/GroupWise_Object_API) page on the NDK and viewed at MAIN.CPP (../../../samplecode/gwobjapi_sample/QUERY/CPP/MAIN.CPP.html).

6.9.1 Object API Objects

This application uses the following Object API objects:

Account
Application
Folder
Folders
FormattedText
Message
MessageList
Query

6.10 Screen Saver

Uses the GroupWise Object API to provide a message-enabled screen saver. When the screen saver is running on your computer, someone can stop by your office, type a note in the screen saver, and send the note to you via GroupWise.

This sample can be downloaded from the GroupWise Object API (http://developer.novell.com/wiki/index.php/GroupWise_Object_API) page on the NDK and viewed at GWS.CPP (../../../samplecode/gwobjapi_sample/SCNSAVER/CPP/GWS.CPP.html).

6.11 Shared Folders

Demonstrates how to create shared folders using the Object API.Demonstrates how to create shared folders using the Object API.

This sample can be downloaded from the GroupWise Object API (http://developer.novell.com/wiki/index.php/GroupWise_Object_API) page on the NDK and viewed at SHARE.CPP (../../../samplecode/gwobjapi_sample/SFOLDER/CPP/SHARE.CPP.html).

6.11.1 Object API Objects

This application uses the following Object API objects:

Account
Address
AddressBook
AddressBookEntries
AddressBookEntry
Application
Folder
FolderRights
FolderRightsCollection

A Revision History

The following table lists changes made to the GroupWise Object API documentation (in reverse chronological order):

Release	Changes
November 2012	Reviewed and updated for use with GroupWise 2012.
December 2011	Added PhoneNumber (page 141).
	Added PhoneNumbers (page 142).
	Added CellPhoneCarrierConstants (page 176)
	Added PhoneNumberConstants (page 186).
March 2009	Added Rules Support for GroupWise 8.0 to documentation.
	Made changes to Attachments (page 58).
December 2008	Added additional Methods and Remarks for GroupWise 8.0 functionality to Attachments (page 58).
	Added additional Properties for GroupWise 8.0 functionality to Attachment (page 56).
May 2008	Added Appointment “Methods” on page 44.
	Added Section 4.32, “MessagePropertiesConstants,” on page 185.
	Added MessageProperty Property to Appointment (page 52), DocumentReference (page 77), Mail (page 123), Note (page 136), PhoneMessage (page 139), SharedNotification (page 158), and Task (page 162).
June 2007	Added section on what to use after importing the type library to Section 2.3, “C ++,” on page 21.
	Clarified that a NULL object pointer is returned when an exception is thrown for several of the login methods of the Application (page 49) object.
	Added that the AllDay property was added in GroupWise 7.0.1 to the Appointment (page 52) object.
February 2007	Added the Boolean type to the AllDay property of Appointment (page 52).
	Added a note about a defect in GroupWise 6.5 and earlier versions preventing document references from appearing in a folder’s Messages collection to the Remarks section of Folder (page 101).
	Removed a note from the PlainText property of FormattedText (page 113).

Release	Changes
October 2006	Added a note about needing to be logged in as a Trusted Application to set the backup or retention dates in SOAP for Account (page 26).
	Added the Signature (page 160) object and updated the Signatures (page 161) object.
	Added an explanation and example of the AllDay property to Appointment (page 52).
	Added the Section 5.3.2, “AddressBook Keywords,” on page 201 section.
	Added filter expressions for not read and not opened to Section 5.4.1, “UNARY,” on page 202.
	Added a closing parenthesis to one of the examples in Section 5.7, “Examples of Valid Filter Expressions,” on page 207.
	Added filter expression examples for the address book to Section 5.7, “Examples of Valid Filter Expressions,” on page 207
June 2006	Added a note about how GroupWise 7.0.1 changed the way Personal or System Groups are added to the Recipients (page 150) object of any email or appointment.
	Removed the note about supporting free threading from Section 1.5, “Support and Limitations,” on page 14. It’s best to use Apartment Threading with GroupWise.
March 2006	Added the AccountProperty to Account (page 26) and the corresponding Section 4.1, “AccountPropertiesConstants,” on page 172 section.
	Updated the description of the Stream property of Attachment (page 56).
	Changed the MESSAGE CONTAINS example in Section 5.7, “Examples of Valid Filter Expressions,” on page 207.

Release	Changes
October 2005	Transitioned to revised Novell documentation standards.
	Added a section about not using a Visual Basic packager to install Object API (see Section 1.8, “Visual Basic,” on page 18).
	Updated the description of the Count property of AddressBookEntries (page 39).
	Added the BuildNumber and VersionRevision properties to Application (page 49).
	Added the AllDay property to Appointment (page 52).
	Added the Size property to Appointment (page 52), DocumentReference (page 77), Mail (page 123), Note (page 136), PhoneMessage (page 139), SharedNotification (page 158), and Task (page 162).
	Added the Stream property and Example section to Attachment (page 56).
	Added a note about searching only on First name and Last Name in the Find method of AddressBookEntries (page 39).
	Added a note about the name of the root folder changing for GroupWise 7.0 to the description of the Name property of Folder (page 101).
	Updated the description of the Delete method in FieldDefinition (page 95). Delete will not delete a field definition that is associated with the AddressBook object.
	Added a note about Unicode characters to the PlainText property description of FormattedText (page 113).
	Added the egwEmailAddressTokenAPI value to Section 4.19, “EmailAddressFormatConstants,” on page 180.
	Removed the AddressBookEntries2 and Folders2 objects.
March 2005	Added AddressBookEntries2 and Folders2.
	Added the AccountUID, GetQuickMessagesCollectionExt, and GetSystemAddressBook methods. Also, added the PeekModeFlag and AccountUID properties to Account (page 26).
	Added the GetQuickMessagesCollectionExt method and the PeekModeFlagConstants to Folder (page 101).
	Added the ClassName and StartDate properties to QuickMessage (page 145).
	Added the Archived property to PhoneMessage (page 139).
	Updated the values of Section 4.37, “QuickMessagesCreationConstants,” on page 187.

Release	Changes
October 2004	Added the QuickMessage (page 145) and QuickMessages (page 147) objects. Added the GetQuickMessagesCollection to Account (page 26) and Folder (page 101). Added the following properties: ◆ AlarmReminderMinutes and AlarmSet to Appointment (page 52). ◆ AttachmentSize and ModifiedDate to Attachment (page 56) ◆ ModifiedDate to Folder (page 101) Added Section 4.37, “QuickMessagesCreationConstants,” on page 187 and added value 7 egwQuickMessageParent to Section 4.11, “CategoryParentObjectTypeConstants,” on page 175. Moved the Archived and Sharer properties and the LocalDelete method from Message (page 125) to Appointment (page 52), DocumentReference (page 77), Mail (page 123), Note (page 136), SharedNotification (page 158), and Task (page 162). Added a note about the Add method creating folders only at the current folder level to Folders (page 109). Fixed minor spacing problem in Revision History entry and added navigational links.
June 2004	Added a note about a memory leak with the Add method of Fields (page 98) and the Add method of AddressBooks (page 45). Updated the description of the DistinguishedName, LastRetentionDate, and LastBackupDate properties of Account (page 26). Added more details to the description of the EmailType property of Recipient (page 148). Added a note about the FindMessages method of Folder (page 101) being asynchronous. Updated the Bowling Shoe Size example in Section 5.3, “Keywords,” on page 195.
February 2004	Updated the description of the EMailType property of Recipient (page 148). Added examples of the Add method to Recipients (page 150).
December 2003	Added links to downloadable and viewable samples in Chapter 6, “Sample Applications,” on page 209. Added information about the format of the StartDate property to the Remarks section of Appointment (page 52). Updated the samples of TASK_CATEGORY in “Parser Numeric Statement” on page 16. Added the UsingSSL property to Account (page 26), the MapiEntryID property to AddressBookEntry (page 41), the MultiLoginAddressBookSupport property to Application (page 49), and the CollectionParent proprety to DocumentType (page 82). Removed the MajorVersionNumber and MinorVersionNumber properties from Account (page 26).

Release	Changes
October 2003	Added documentation for Section 1.7.7, “Date Fields,” on page 17.
	Updated the description of the MultiLogin method of Application (page 49), the Stop method of Query (page 143), the Add method of Messages (page 131), and the EMailAddresses property of AddressBookEntry (page 41).
	Updated the Folder (page 101) object.
	Updated the description of the CATEGORY keyword in “Available Keywords” on page 198.
	Added LastBackupDate, LastRetentionDate, and PeekMode properties to Account (page 26). Added DeliveredDate, Forwarded, and Replied properties to Appointment (page 52), DocumentReference (page 77), Mail (page 123), Note (page 136), PhoneMessage (page 139), SharedNotification (page 158), and Task (page 162). Added egwAllowPasswordPrompt to Section 4.29, “LoginConstants,” on page 184.
	Added three birthday fields to AddressBookEntry (page 41).
June 2003	Updated the name of EMailAddresses (page 93).
March 2003	Updated the documentation for GroupWise 6.5 features (see additional Section 4.26, “FolderTypeConstants,” on page 183).
	Added documentation for Categories (page 65), Category (page 66), CategoryDefinition (page 67), CategoryDefinitions (page 68), DraftAutoDates (page 91), EMailAddress (page 92), EMailAddresses (page 93), IMAddress (page 118), and IMAddresses (page 119).
	Updated Chapter 5, “Filter Expressions,” on page 195.
September 2002	Updated the Preface.
May 2002	Added information about the AddExistingMessage method to Messages (page 131).
	Updated the SetPassword and SynchronizeToRemote methods in Account (page 26). Updated the AddExistingMessage method in Messages (page 131).
February 2002	Updated the FindMessage method of Folder (page 101).
	Restored Archived and Sharer properties as well as the LocalDelete method to Message (page 125).
	Added the AddExistingMessage method to Messages (page 131), as well as constants for this method, Section 4.33, “MessageStatusConstants,” on page 185 and Section 4.34, “MessageSecurityConstants,” on page 186.
	Added Section 4.46, “SynchronizeConstants,” on page 190.
	Corrected example for Basic Expressions.
October 2001	Added two functions to Account (page 26): SynchronizeToRemote and SynchronizeWithMaster.
	Updated the AddEx function description for AddressBooks (page 45).
	Removed the Archived and Sharer properties from Message (page 125) since they never worked properly. Also, for the same reason, removed the LocalDelete method from Message.

Release	Changes
September 2001	Added support for GroupWise 6 to documentation.
	Added the SetPassword function to Account (page 26).
	Added a statement to AddressBookEntries (page 39) that the Add function can be called only when adding an address book entry to a personal address book.
	Added a statement to GroupMembers (page 115) that AddressBookEntry objects must have a valid email address.
	Added text alternatives to figures.
June 2001	Added addditional information about the DisplayName property in AddressBookEntry (page 41), the Count property in AddressBookEntries (page 39), the PathToArchive property in Account (page 26), and the Add method in Documents (page 80).
	Changed the Attachment collection to the Attachments collection in Message (page 125).
	Removed the ChangeOwner method of Folder (page 101) since the method is not yet implemented.
	Added an explanation of a valid AddressBookEntry to the Item method of FolderRightsCollection (page 107).
	Added an explanation of filtering on five fields using the GroupWise System AddressBook to Filter Expression Syntax.
	Added an explanation of using Windows NT service to the “About This Guide” on page 9.
	Improved document accessibility.
February 2001	Revised documentation for early and late binding using Visual Basic. Added instructions in Visual Basic on how to import a type library. Also, moved task-oriented information to a separate Tasks chapter.
	Added a note to “About This Guide” on page 9 about the Client/Cache mode and the offline database.
September 2000	Documented the 32KB limit for message body text.
July 1998	Documentation added to the Novell Developer Kit.