MICRO FOCUS

GroupWise® Developer Kit

Tokens

March 2020

Legal Notices

The only warranties for products and services of Micro Focus and its affiliates and licensors (“Micro Focus”) are set forth in the express warranty statements accompanying such products and services. Nothing herein should be construed as constituting an additional warranty. Micro Focus shall not be liable for technical or editorial errors or omissions contained herein. The information contained herein is subject to change without notice.

Contents

About This Guide

The GroupWise Token APIs and Tokens let you manipulate the GroupWise client interface by subscribing to internal token events or by publishing new tokens to the client. For example, using tokens, you can manipulate message text, change client settings, display client dialog boxes, and automatically trigger events such as archive and empty trash.

IMPORTANT: Unless otherwise marked, the features in GroupWise Token APIs will work 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 Documentation

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

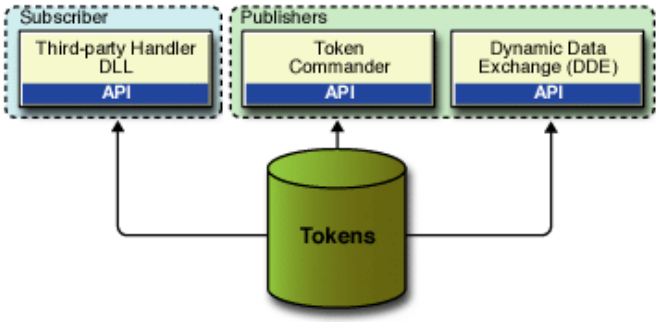
1 Overview

This section provides overview information for GroupWise Token APIs and Tokens and covers the following topics:

GroupWise Tokens is comprised of the following:

- One Subscribing API: Third-Party Handler DLL API
- Two Publishing APIs:
  - Token Commander API
  - Dynamic Data Exchange (DDE) API
- Tokens are low-level events that are packaged into a meaningful representation and are subscribed or published by one of the Token APIs as illustrated in the graphic below.

Figure 1-1 *Tokens Architecture*



There are more than 600 tokens that act as powerful tools to help you extend the functionality of your GroupWise implementation.

Token APIs and Tokens Defined

See the following topics in this section:

Why Use Tokens

The GroupWise Windows client uses tokens to perform various different internal functions. A tokenized application is one where low-level events are packaged into a meaningful representation or tokens, such as *Save the current message to a file* or *Create a new mail message*. These tokens are often processed by the application and become the basis for everything the application does.

The GroupWise client must be running for a token to be processed. If the client is not running, the Token Commander API will start up the client and then process the token. However, on some operating systems the token can be processed before the client is fully running, resulting in token failure. This problem can be solved in one of two ways:

- Make sure that the GroupWise client is fully up and running before trying to access the Token Commander
- Provide a delay in the application code after calling the Token Commander (such as providing a "sleep" loop) before throwing any tokens

What Is a Token

A *token* in GroupWise is used to represent a structure of information. The token structure contains numerous properties, including

- the token ID (the token identifier)
- the token parameter count
- the token parameters specific to the token ID

Note that the term *token* and *token ID* often refer the same thing - the token ID. The token ID is a numeric representation for the functionality the token structure relates to. Third-party applications can use the token identifier or the numeric value when working with the token ID. For example, the token used to display a new appointment message view is `NewAppointment()`. The numeric token identifier is `BFTKN_SEND_STDAPPT` or the decimal number 182. The term `NewAppointment()` is the text representation of the token ID. To maintain future compatibility, it is recommended that the numeric token identifier is used instead of the numeric value in the event the numeric values change in the future. The token structure is defined as follows:

```
typedef struct _tagMAC_TOKEN
{
        HSZ hszCommand;
        MAC_IPCVERSION Version;
        HSZ hszRequestor;
        MAC_MACROID dwMacroID;
        ATOM atomApp;
        WORD wReserved;
        MAC_TOKENID wTokenId;
        MAC_COUNT cParam;
        DWORD dwFlags;
        DWORD dwReserved;
        MAC_PARAM rgParam[1];
}
```

How Tokens Work

Tokens are published internally as a result of low level events such as a mouse click or a key being pressed. These tokens are analogous to Windows messages (i.e. WM_SIZE). Tokens are also published as a result of higher level events like a user replying to a message. Whether or not a token

works properly depends upon the context when the token is published. For example, the `Type()` (`AFTKN_TYPE`) token can be used to simulate the keyboard to type some text. If a message view is opened and the focus is on the message body field, then the context is valid because you can type text in the message body field. If the focus is on the list of messages in your mailbox, publishing the `Type()` token will return an error because the context is invalid. There is nothing to receive the text the token will *type*.

Another example of how a token works is when you press the reply button, the client will publish the `ItemReply` (`BFTKN_REPLY`) token. The token router functionality in the client will call the proper token handler to process the token. This token handler causes the reply message view to be displayed and the proper information inserted into the view. In the `ItemReply()` token, the token parameters contain information on whether the reply is to go to everyone or just the sender of the original message, and whether or not the original message text is inserted into the reply message.

Token Characteristics

See the following topics in this section:

- "User Interface Tokens" on page 21
- "Tokens and Message ID" on page 21

User Interface Tokens

When dealing with tokenized applications, it is sometimes useful to divide the available tokens into two classes:

- `UI tokens` relate to some action on the screen. For example, the `AddressBookDlg()` (`DTKN_ADDRBOOK`) token displays the address book dialog on the screen. UI tokens usually need user intervention.
- `UI-less tokens` produce no visible affect on the screen. For example, the `PrefAdvanced` (`AFTKN_SET_ADV`) token can programmatically set properties of a message view without having to open the message view properties dialog. UI-less tokens provide functionality without user intervention.

Tokens and Message ID

Tokens often require a message ID as a parameter. A message ID is a string that uniquely identifies a GroupWise message. Every message in the GroupWise account contains a message ID. Many tokens use message IDs to manipulate messages. For example, the message ID is required when you need to open a message or when you need to get attributes from a message. To open a message, the `ItemOpen("39255B499091137016")` (`AFTKN_ITEM_OPEN`) token opens the message identified by the message ID 39255B499091137016.

Message IDs are persistent. Third-party applications are free to store message IDs external to GroupWise. Note that a message will have one message ID when it is created in the GroupWise remote client, then when it is synchronized with the master mailbox, it will receive a new message ID. It should also be noted that when a third-party application stores a message ID, after long periods of time (i.e. the message has been deleted) or exceptional system events (i.e. moving the user to a new post office), a message ID may no longer be valid. In these situations, the stored message ID should be discarded.

A Message ID is a string created by GroupWise that does not have any obvious meaning in and of itself to third-party applications other than to be used as a way to uniquely identify messages. When using a message ID, the third-party application should not manipulate the message ID other than to pass the string as a parameter in a token.

The messageID structure is used only by the MAPI provider code, so it cannot be used with Object API or GroupWise Tokens.

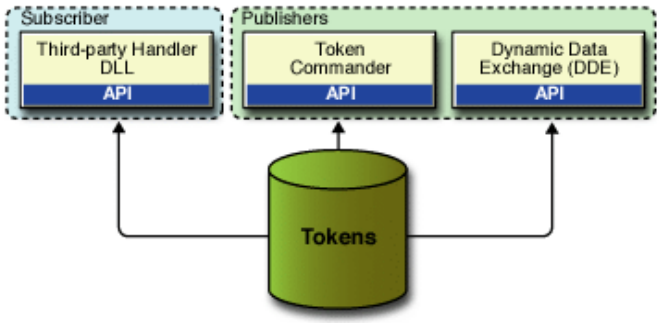
Message ID X00

When you create a new message in the GroupWise client, the message ID has not yet been assigned. The message ID is assigned after the user has pressed the send button and the message view has disappeared. In order to publish tokens for message views that have not yet been sent, GroupWise uses *X00* as a temporary message ID for the current displayed message. This is different than the Object API where a draft message is always created with a valid message ID. Tokens do not create draft messages. Likewise, the Object API does not use the temporary *X00* message ID.

Subscribe and Publish

GroupWise Tokens is comprised of three APIs that subscribe and publish the tokens, as shown below.

Figure 1-2 *Subscribe and Publish Tokens*



The APIs include:

- One Token Subscribing API: Third-Party Handler DLL API
- Two Token Publishing APIs:
  - Token Commander API
  - Dynamic Data Exchange (DDE) API

All three of the APIs use tokens at the core. Tokens are subscribed by or published by one of the token APIs.

Token Subscribing

Subscribing to tokens is the process whereby a third-party application is given a token as it is published before the GroupWise default token handler processes the token. The Token Commander and DDE are ways to publish token events.

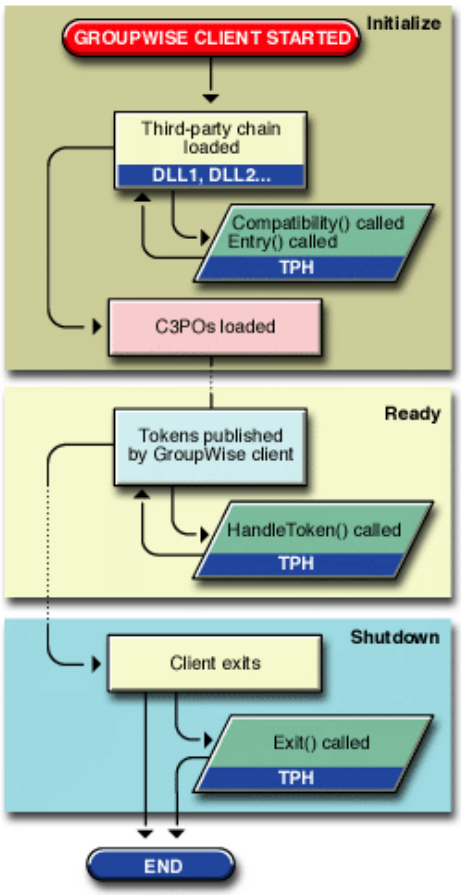
Third-Party Handler DLL API

The TPH DLL API allows you to enhance the user's experience with the GroupWise client by allowing third-party applications to subscribe to internal token events. The TPH DLL API is the mechanism that allows a third-party application DLL to subscribe to token events. A TPH DLL can monitor the token stream listening for tokens that are of interest to the application. TPH DLLs are automatically loaded by the GroupWise client at startup. TPH DLLs are loaded before GroupWise C3PO DLLs are loaded.

The set of all TPH DLLs for a particular workstation are referred to as the third-party DLL chain. The DLLs in the third-party chain have the opportunity to process a token before it is processed by GroupWise. Tokens are always first passed to the third-party chain where any TPH DLL can modify, remove, or ignore any given token. If a TPH DLL modifies or removes a token, all subsequent TPH DLLs in the chain will receive the modified token, or will not receive the deleted token. After all TPH DLLs in the third-party chain have evaluated the token, if it is still in the token stream GroupWise will process the resulting token.

The following figure illustrates the behavior of TPH during initialization, operation, and shutdown of the GroupWise client.

Figure 1-3 *TPH Behavior*



TPH Interface Requirements

The TPH DLL Interface requires the following:

1. A Windows Registry entry must be added informing the GroupWise client that the TPH DLL wants to be added to the third-party chain.

```
Key
"HKEY_CURRENT_USER\Software\Novell\GroupWise\Client\Third Party"

Item Name
"DLLx" (where x is a number starting at 1)

Item Type
STRING

Item Data
<DLL path and file name>
```

When inserting a TPH DLL entry into the registry, care should be taken to not overwrite any existing TPH DLL entries already registered. Do not always assume your TPH DLL will be DLL1.

You can also register your TPH DLL under HKEY_LOCAL_MACHINE (in addition to HKEY_CURRENT_USER). The full path and contents of the registry entry are the same for HKEY_LOCAL_MACHINE. This allows a TPH DLL to be registered once per machine (rather than requiring it to be registered for each user on a specific machine).

DLLS that were registered using HKEY_CURRENT_USER get priority in the call chain over those that were registered with HKEY_LOCAL_MACHINE. For example, if DLL1 and DLL2 were each registered under HKEY_CURRENT_USER, they would both be called before anything registered under HKEY_LOCAL_MACHINE.

DLLs that were registered using HKEY_LOCAL_MACHINE use the same naming convention for the reg entries (DLL1, DLL2, etc.,) regardless of what might be registered under HKEY_CURRENT_USER. (One DLL might be registered as DLL1 under HKEY_CURRENT_USER and another DLL might be registered as DLL1 under HKEY_LOCAL_MACHINE.)

2. The following platform specific requirements must be met:
   - For C++ DLLs, the C++ Structure Member Alignment must be set to *1-byte alignment*. On some compilers *8-byte alignment* is the default. The GroupWise client was written in C++ and was compiled with 1-byte alignment. If the TPH DLL alignment is not 1-byte, you can experience random memory errors in the Kernel32.dll, and the token data structures will not contain discernible data.

     In C++, make sure the methods are declared with the WINAPI calling convention. For example, the Exit method would be written as: `WORD WINAPI Exit(void)`
   - In Delphi, make sure the methods are declared with the stdcall calling convention. For example, the Exit method would be written as: `Function Exit: SmallInt; export; stdcall;`
   - Visual Basic and other programming languages that can create Win32 DLLs, can be used to create TPH DLLs.

The DLL is required to export seven methods in a specific order to allow GroupWise to load the DLL. The TPH DLL entry points are called using their ordinal values.

The methods are to be exported in the following order:

Export Order	Description
1. TPHVersion	Reserved. Even though it is currently not called by GroupWise, it must be included.
2. Compatibility	Called during the initialization process. It passes the GroupWise client version to the TPH via this method. You can determine the version by evaluating the high and low order bytes of the `AppVersion` parameter.
3. TimeStamp	Reserved. Even though it is currently not called by GroupWise, it must be included.
4. Entry	Used to initialize your TPH specific data. The GroupWise client language is also passed as a parameter to the TPH via this method. You can determine the language by evaluating the high and low order bytes of the `wLanguage` parameter.
5. Exit	Called during the shut down process. It is where you can clean up any memory you allocated.
6. HandleToken	Called whenever a token is published by the GroupWise client. The token is then passed as a parameter.This is the heart of the TPH process. By evaluating the token parameter, you can determine what token was published as well as any token related data in the token structure.
7. ValidateToken	Reserved. Even though it is currently not called by GroupWise, it must be included.

Token Publishing

Publishing tokens is the process whereby a token is passed to the GroupWise client to cause it to act upon the token. The TPH DLL API is where an application subscribes to token events.

- "The Token Commander API" on page 25
- "DDE" on page 26

The Token Commander API

The Token Commander API allows you to publish tokens to the GroupWise client. This API is COM-based and replaces the older DDE interface for publishing tokens. The DDE interface for tokens is still supported in both the 16-bit and 32-bit GroupWise 5 clients. See "DDE" on page 26 for more information.

The Token Commander interface is simple and contains only one method: `Execute()`. The syntax for this method uses the text version of the token. For example, to open a specific message in the client, the application publishes the `ItemOpen` token similar to this

```
Execute("ItemOpen(\"39255B499091137016\")")
```

GroupWise then takes the token and passes it through the third-party token chain for any TPH DLL that wanted to subscribe to the token to process it.

Other examples include publishing tokens to manipulate message text with the `ItemSetText()` (`AFTKN_ITEM_SET_TEXT`) token, change client settings with the `PrefEnvironment` (`AFTKN_SET_ENV`) token, display client dialog boxes with the `FilterDlg` (`DTKN_FILTER_ITEMLIST`) token, and automatically trigger events such as archive a message with the `ItemArchive` (`AFTKN_ITEM_ARCHIVE`) token.

DDE

Simply put, Dynamic Data Exchange (DDE) helps Windows applications communicate. Applications establish a DDE conversation and send messages to each other. A DDE conversation involves *server* and *client* applications. The server provides resources to the client. The client accepts processing or information from the server.

Your GroupWise solution can:

- Share GroupWise resources using DDE Execute.
- Obtain GroupWise processing results using DDE Request.

DDE Management Library (DDEML)
The *DDE Management Library (DDEML)* simplifies DDE management by providing a high-level interface. This document discusses DDEML and not lower-level DDE calls. Use `DdeInitialize()` to register an application with DDEML.

DdeConnect()
Use `DdeConnect()` to establish a conversation. A service parameter specifies an application to establish a conversation with. A topic parameter specifies the conversation topic. See “Conversation with GroupWise” on page 26.

DdeClientTransaction()
Use `DdeClientTransaction()` to send service and information requests between conversing applications.

DdeDisconnect()
Use `DdeDisconnect()` to end a conversation.

DdeUninitialize()
Use `DdeUninitialize()` to clear application registration.

Conversation with GroupWise

A DDEML-registered application can establish a conversation with GroupWise using `DdeConnect()` including a service and topic name pair from the table below. The first pair is generally recommended:

Service Name	Topic Name	Comments
GroupWise	Command	Recommended. A conversation using this pair can send any supported command to GroupWise.
Foreign	Command	Operates like GroupWise/Command but makes GroupWise compatible with applications that launch a server based on the service name, in this case, OFWIN.EXE.

For example:

```
const char lpszService[ ] = "OFWIN";
const char lpszTopic[ ] = "COMMAND";

HDDEDATA FAR PASCAL __export DdeCallback( WORD, WORD, HCONV, HSZ, HSZ,
HDDEDATA, DWORD, DWORD );

DWORD dwDDEInst;
HSZ   hszService;
HSZ   hszTopic;
HCONV hConv;

/** Register with DDEML **/
DdeInitialize(&dwDDEInst,(PFNCALLBACK)DdeCallback, APPCLASS_STANDARD |
   APPCMD_CLIENTONLY, 0L);

/** Create our String Handles **/
hszService = DdeCreateStringHandle(dwDDEInst, lpszService, 0);
hszTopic = DdeCreateStringHandle(dwDDEInst, lpszTopic, 0);

/** Attempt to establish connection **/
hConv = DdeConnect(dwDDEInst, hszService, hszTopic, NULL);

/** Free the string handles **/
DdeFreeStringHandle(dwDDEInst, hszService);
DdeFreeStringHandle(dwDDEInst, hszTopic);
```

Sending GroupWise Commands

Your solution can send GroupWise token in a DDE Execute parameter.

When GroupWise cannot process a command, it stores user-accessible error information. See “Using DDE to Get Error Information” on page 28.

DDE Execute String Format

DDE Execute strings are null-terminated ANSISTRINGs. Each string consists of supported tokens. You can space-delimit multiple tokens in a string. For more information, see the GroupWise 4.1 Macros Manual in the GroupWise 4.1 Developer Components download at http://developer.novell.com/ndk/unsupported.htm (http://developer.novell.com/ndk/unsupported.htm).

For example:

```
"ViewOpenDlg( )"
"ViewOpen( Appointment!; Yes! )"
"ViewOpen( Appointment!; Yes!) FocusSet( Message! ) EditPaste( )"
"ViewOpen(Appointment!;Yes!)FocusSet(Message!)EditPaste()"
"ViewOpen(Appointment!;Yes!) FocusSet(Message!) EditPaste( ) "
```

DDE ignores the remainder of a transaction containing an invalid command or syntax. For more information, see the GroupWise 4.1 Macros Manual in the GroupWise 4.1 Developer Components download at http://developer.novell.com/ndk/unsupported.htm (http://developer.novell.com/ndk/unsupported.htm).

Send GroupWise a token string as the DDE data block in a DDE Execute transaction. This is the Windows `DDEClientTransaction()`, `lpvData` parameter.

For example:

```
/** lpszCmd is a pointer to an ANSISTRING **/
/** containing the token(s) we wish to **/
/** send to GroupWise **/
DdeClientTransaction(
   (LPBYTE)lpszCmd,
   (DWORD)_fstrlen(lpszCmd)+1,
   hConv,
   NULL,
   CF_TEXT,
   XTYP_EXECUTE,
   (DWORD)TIMEOUT_ASYNC,
   NULL );
```

Using DDE to Get Error Information

Your solution can use DDE to request GroupWise error information. `DDEGetLastError()` returns information about the cause of the last error when `DDEClientTransaction()` returns FALSE.

A DMLERR_NOTPROCESSED return value indicates GroupWise has received a token but cannot process it.

Send an `EnvLastError()` DDE Request to GroupWise to return an error information string. Successful GroupWise tokens empty the error value so that `EnvLastError()` returns an empty string. The following table lists and describes error values:

Value	Description
10E1	Bad Parameter x, where x is the index of the bad parameter.
10E2	Failed.
10E3	Command is invalid given current state of GroupWise.
10E4	Not found.
10E5	Syntax error.
10E7	BC (Blind Copy) does not apply to personal items.
10E8	Not supported for encapsulated item attachments.
10E9	Not supported for OLE attachments.
10EA	The message ID references an Out Box message that does not exist yet.
10EB	You cannot set an alarm for a time that has already expired.
10EC	The source attributes of the In Box filter cannot be changed.
10ED	The source attributes of the Out Box filter cannot be changed.
XXXX	Other GroupWise error.

For example:

```
HSZ hCmd;

/** Create a valid string handle for GroupWise Token **/
hCmd = DdeCreateStringHandle (dwDDEInst, "EnvLastError()", CP_WINANSI);

/** Request the information **/
if ( 0 != (hReqData = DdeClientTransaction( NULL, (DWORD)0L, hConv,
   hCmd, CF_TEXT, XTYP_REQUEST, (DWORD)2000, NULL )))
{

   /** Free our string handle **/
   DdeFreeStringHandle (dwDDEInst, hCmd);

   /** Get a valid pointer to the data that we can access **/
   lpDataByte = DdeAccessData (hReqData, &cbDataLen);

   /** Copy the data into the return buffer */
   lstrcpy (lpszBuf, (LPSTR)lpDataByte);

   /** THE ERROR MESSAGE IS NOW STORED IN lpszBuf **/

   /** Free up the pointer we locked down **/
   DdeUnaccessData (hReqData);
   return TRUE;
}
```

Requesting GroupWise Information

To obtain GroupWise return value information, use:

- EnvLastCmdResult token
- DDE Request
- DDE Advise Loop

GroupWise returns information in ANSISTRINGs. You can convert returned ANSISTRING data to its native data type. For example, GroupWise returns a numeric value 96 as the ANSISTRING 96. Native data types and corresponding GroupWise ANSISTRINGs include:

Native Data Type	ANSISTRING Returned by GroupWise
Numeric	Numeric characters
Boolean	"TRUE", "FALSE"
ANSISTRING	ANSISTRING

EnvLastCmdResult

You can request GroupWise DDE Execute return information using DDE Request including `EnvLastCmdResult()`. For example:

```
HSZ hCmd;

/** Create a valid string handle for our command string **/
hCmd = DdeCreateStringHandle (dwDDEInst, "EnvLastCmdResult()",
CP_WINANSI);

/** Request the information **/
if ( 0 != (hReqData = DdeClientTransaction( NULL, (DWORD)0L, hConv,
hCmd, CF_TEXT, XTYP_REQUEST, (DWORD)2000, NULL)))
{
   /** Free our string handle **/
   DdeFreeStringHandle (dwDDEInst, hCmd);

   /** Get a valid pointer to the data that we can access **/
   lpDataByte = DdeAccessData (hReqData, &cbDataLen);

   /** Copy the data into the return buffer */
   lstrcpy (lpszBuf, (LPSTR)lpDataByte);

   /** THE COMMAND RESULT DATA IS NOW STORED IN lpszBuf **/

   /** Free up the pointer we locked down **/
   DdeUnaccessData (hReqData);
   return TRUE;
}
```

DDE Request

DDE Request substitutes DDE Execute command strings for DDE Item names and obtains return values. DDE Request returns a DDE data handle, then lets your solution have it.

For example:

```
HSZ hCmd;
/** Create a valid string handle for our command string **/
hCmd = DdeCreateStringHandle (dwDDEInst, "AddressBookGetField(
ABField:
   USERSNETID! )", CP_WINANSI);
/** Request the information **/
if
   ( 0 != (hReqData = DdeClientTransaction( NULL, (DWORD)0L, hConv,hCmd,
   CF_TEXT, XTYP_REQUEST, (DWORD)2000, NULL)))
{
   /** Free our string handle **/
   DdeFreeStringHandle (dwDDEInst, hCmd);

   /** Get a valid pointer to the data that we can access **/
   lpDataByte = DdeAccessData (hReqData, &cbDataLen);

   /** Copy the data into the return buffer */
   lstrcpy (lpszBuf, (LPSTR)lpDataByte);

   /** THE COMMAND RESULT DATA IS NOW STORED IN lpszBuf **/

   /** Free up the pointer we locked down **/
   DdeUnaccessData (hReqData);

   return TRUE;
}
```

DDE Advise Loop

You can establish a DDE Advise loop instead of using `EnvLastCmdResult()`. A DDE Advise loop returns DDE Execute values to DDE Advise Data, then GroupWise, notifies your solution. DDE notifies your solution only of DDE Advise Data values for commands that return values and commands sent to GroupWise not using DDE Request.

```
HSZ hItem;
HCONV hConv; // These two variables are
DWORD
dwDDEInst; // assigned valid values
// during DDE initialization

/** Create a valid string handle for our command string **/
hItem =
DdeCreateStringHandle (dwDDEInst, "CommandReturn", CP_WINANSI);

/** Start the Advise Loop **/
DdeClientTransaction( (LPBYTE)NULL, 0, hConv, hItem, CF_TEXT,
   XTYP_ADVSTART, (DWORD)TIMEOUT_ASYNC, &dwTransResult
);
```

Simultaneous Commands

DDE does not let your solution send DDE Execute during another same-conversation command. Make sure your solution does not send simultaneous multiple DDE Execute or DDE Request commands. Otherwise, a DDE Reentrancy error results and your commands may not execute.

Data Types in This Documentation

This topic lists the data types most commonly used in the GroupWise Token functions. All of the data types are exactly the same as those in the Windows Software Development Kit (SDK).

- ANSISTRING—A 32-bit pointer to a character string.
- DWORD—A 32-bit unsigned integer
- long—A 32-bit signed integer
- WORD—A 16-bit unsigned integer
- int—A 16-bit signed integer
- BYTE—A 8-bit unsigned character
- char—A 8-bit signed character
- BOOL—A Boolean value (True / False)

2 Tasks

This section provides information about how to design and develop applications that take advantage of GroupWise Token APIs and Tokens.

This section covers the following:

Design

There are two main questions you want to ask yourself before starting: What do you want to accomplish and will one of the Token APIs accomplish this? The Token APIs are client-based, which means that the client has to be running in order to use the APIs. Tokens drive the client, so if you want to subscribe to client events or publish client events in a certain order, then the Token APIs are what you want to use. If you want to work only with the GroupWise account data, then you might need to use the Object API.

If you want to subscribe to token events so you can alter the client behavior, then you need to use the Third-Party Handler (TPH) DLL API. To do this, you first need to identify what tokens you want to subscribe to. The TPH, by design, will receive all external tokens as they occur in the GroupWise client. The tokens are passed via the `HandleToken()` method that all TPH DLLs are required to support. The TPH will need to evaluate the token ID to see if the it is one the TPH wants to handle.

In the example, we want to stop the end user from modifying the GroupWise toolbar. In addition to removing this functionality from the client, we want to add alternate functionality by displaying the GroupWise about dialog instead of the edit toolbar dialog. For the toolbar token we want to subscribe to the `DTKN_BTNBAR_EDIT` token ID. For the about dialog token, we will publish the `AboutDlg()` (`DTKN_ABOUT`) token.

Sample Code

This section covers the following:

To create a valid TPH, we are required to have seven methods in our DLL even though some of them are not used by the GroupWise client. The reason the unused methods are required is because the methods that are used, are called from the GroupWise client by their exported order number. This is why they are required to be exported in a specific order. In C++, this is easily accomplished in the.DEF file. The following is a sample from a C++ .DEF file and also identifies the order in which the methods must be exported:

```
EXPORTS
     TPHVersion     @1
     Compatibility  @2
     TimeStamp      @3
     Entry          @4
     Exit           @5
     HandleToken    @6
     ValidateToken  @7
```

In Delphi, you need to use the following code in the file that contains the TPH methods:

```
exports
     TPHVersion     index 1 resident,
     Compatibility  index 2 resident,
     TimeStamp      index 3 resident,
     Entry          index 4 resident,
     Exit           index 5 resident,
     HandleToken    index 6 resident,
     ValidateToken  index 7 resident;
begin
end.
```

Below are skeletons of source code for all the required TPH methods, written in C++ and Delphi formats. These skeletons will be the basis for our example and we will add functionality to them as we go.

C++ Skeleton

For C++, note that all the methods require the WINAPI specifier before the method name:

```
DWORD WINAPI TPHVersion ( void )
{
     return 0;
}

DWORD WINAPI Compatibility( ATOM AppAtom, WORD AppVersion)
{
     // Greater than or equal to GroupWise 5.2
     if (5 >= LOBYTE(AppVersion) && 2 >= HIBYTE(AppVersion))
     {
          return (DWORD) TRUE;
     }
     else
     {
          return (DWORD) FALSE;
     }
}


DWORD WINAPI TimeStamp( void )
{
      return 0;
}
```

```
WORD WINAPI Entry( WORD wLanguage)
{
     if ('U' == (char) LOBYTE(wLanguage) && 'S' == (char)HIBYTE(wLanguage))
     {
          return 1;
     }
     else
     {
          return 0;
     }
}


WORD WINAPI Exit( void )
{
     return 1;
}

int WINAPI HandleToken(LPTPH_RETURNVAL lpTokenData, HWND hLinkWnd, WORD msg )
{
     return DLL_HAN_NOT_HANDLED;
}


DWORD WINAPI ValidateToken(LPMAC_TOKEN lpTokenData, Lpvoid lpDataTypeInfo)
{
      return DLL_VAL_UNKNOWN_TOKEN;
}
```

Delphi Skeleton

For Delphi, note that all the functions require the export and stdcall specifier after the parameter list:

```
Function TPHVersion: LongInt;export; stdcall;
begin
     result := 0;
end;

Function Compatibility(AppName: ATOM; AppVersion: Word): LongInt; export; stdcall;
begin
     if (5 >= LO(AppVersion) and 2 >= HI(AppVersion) then
     begin
          result := 1;
     end
     else
     begin
          result := 0;
     end;
end;

Function TimeStamp: LongInt; export; stdcall;
begin
     result := 0;
end;

Function Entry(wLanguage: Word): SmallInt; export; stdcall;
begin
     if ( 'U' = Chr(LO(wLanguage)) and 'S' = Chr(HI(wLanguage)) then
     begin
```

```
          result  := 1;
     end
     else
     begin
          result  := 0;
     end;
end;

Function Exit: SmallInt; export; stdcall;
begin
     result := 1;
end;

Function HandleToken(lpTokenData: LPTPH_RETURNVAL; hLinkWnd: HWND; msg: Word):
SmallInt; export; stdcall;
begin
     result := SmallInt(DLL_HAN_NOT_HANDLED);
end;

Function ValidateToken(lpTokenData: LPMAC_TOKEN; lpDataTypeInfo: LPVOID): LongInt;
export; stdcall;
begin
     result := 0;
end;
```

Registering A TPH

GroupWise client TPH DLLs must be registered in the Windows registry so the GroupWise client knows where to find the DLL. The Windows registry location and settings are as follows:

```
Key
"HKEY_CURRENT_USER\Software\Novell\GroupWise\Client\Third Party"

Item Name
"DLLx" (where x is a number starting at 1. Make sure you do not overwrite an
existing entry)

Item Data
 <DLL path and file name>

Item Type
STRING
```

Initialization of A TPH

When a TPH is properly registered in the Windows registry, and the client is started, GroupWise loads the TPH DLLs listed in the registry. For each TPH it loads, it calls the `Compatibility()` and `Entry()` methods. Note that all TPH DLLs are loaded before C3POs are loaded.

The `Compatibility()` method lets a TPH specify the GroupWise version that can load it and ensures compatibility between GroupWise and TPH DLLs. The TPH can verify major and minor GroupWise client versions, then return to GroupWise a value indicating whether to unload the TPH or not.

The `Entry()` method is where your TPH sets up its necessary data structures, memory allocations, performs initializations, etc. It also tells you the language under which GroupWise is running. Like the `Compatibility()` method above, if the TPH does not support this language, the TPH can then return to GroupWise a value indicating it should unload the TPH.

Subscribing To Tokens

This section covers the following:

As mentioned earlier, all tokens are passed to the TPH method `HandleToken()`. This is the heart of the TPH. The `HandleToken()` method contains three parameters. The first parameter is the token data structure. It contains all the token specific information about the token being published. The second parameter is the handle to the link window. This is always the GroupWise client. The third parameter is the link window execute message. This is the message the client received that caused it to publish the token. This message has no meaning in the client.

The following lines of code are added to the `HandleToken()` method. Here we want to "trap" the toolbar properties token `DTKN_BTNBAR_EDIT`. Because all external tokens are passed to the `HandleToken()` method, we just need to evaluate the `wTokenID` property. If the `wTokenID` was the token we were looking for, we would then call our processing code. In this case, we return the `DLL_HAN_NO_ERROR` value which indicates to GroupWise that we processed the token and that the GroupWise client or other TPH DLLs in the TPH chain don't have process it. In effect it is removed from the token stream.

C++

```
int WINAPI HandleToken(LPTPH_RETURNVAL lpTokenData,
     HWND hLinkWnd,
     WORD msg )
{
     // Is the wTokenID the Toolbar edit token?
     if (DTKN_BTNBAR_EDIT == lpTokenData->lpToken->wTokenId)
     {
          // Return the value indicating we processed the token so the
          // GroupWise client doesn't have to.
          return DLL_HAN_NO_ERROR;
     }

     return DLL_HAN_NOT_HANDLED;
}
```

Delphi

```
Function HandleToken(lpTokenData: LPTPH_RETURNVAL; hLinkWnd: HWND; msg: Word):
SmallInt; export;  stdcall;
begin
     if (DTKN_BTNBAR_EDIT == lpTokenData.lpToken.wTokenId) then
     begin
          result := SmallInt(DLL_HAN_NO_ERROR);
     end;

     result := SmallInt(DLL_HAN_NOT_HANDLED);
end;
```

Now we want to alter the client functionality. Instead of discarding the edit toolbar functionality, we use it to trigger different functionality. The following code will publish the `AboutDlg()` (`DTKN_ABOUT`) token in response to the user selecting the toolbar properties menu. This shows how a TPH can intercept certain events, and inserts custom functionality in between, or instead of the default functionality. Both of the functions below use a custom method called `Publish()` to publish tokens using the Token Commander API. This `Publish()` method is defined in the sample files.

C++

```
int WINAPI HandleToken(LPTPH_RETURNVAL lpTokenData,
     HWND hLinkWnd,
     WORD msg )
{
     LPTSTR lpReturn = NULL; // Used with the Publish method
     if (DTKN_BTNBAR_EDIT == lpTokenData->lpToken->wTokenId)
     {
          // Display the About Dialog instead by publishing a token

          lpReturn = Publish(L"AboutDlg()");
          // If lpReturn = NULL then there was an error,
          // otherwise it will be an allocated string that may or may not
          // contain a value.  You must release it when you are done.
          if ( lpReturn )
          {
               // Evaluate and process the return value here.
               // ...

               // Cleanup
               delete lpReturn;
               lpReturn = NULL;
          }

          // Return the value indicating we processed the token so the
          // GroupWise client doesn't have to.

          return DLL_HAN_NO_ERROR;
     }

     return DLL_HAN_NOT_HANDLED;
}
```

Delphi

```
Function HandleToken(lpTokenData: LPTPH_RETURNVAL; hLinkWnd: HWND; msg: Word):
SmallInt; export;  stdcall;
var
     sResult : String;
begin
     if (DTKN_BTNBAR_EDIT == lpTokenData.lpToken.wTokenId) then
     begin
          sResult := Publish('AboutDlg()');
          if ( sResult <> '' )
          begin
               // Evaluate and process the return value here.
               // ...

          end;
          result := SmallInt(DLL_HAN_NO_ERROR);
      end;

     result := SmallInt(DLL_HAN_NOT_HANDLED);
 end;
```

Include Files

There are various enumerations and data types that the Token APIs use. For C++, the header file is GWDLL.H. For Delphi, the file is GWDLL.PAS. Both files are included in the GroupWise SDK.

In the C++ version of the sample application, you need to link in the Ole32.lib library to take advantage of the `Publish()` method. This method uses the OLE interface into the Token Commander API. You also need to include the following header files:

```
#include "windows.h"
#include "gwdll.h"
#include <initguid.h>
#include <objbase.h>
#include "Ofcmndr.h"
```

3 Third-Party Handler DLL API Reference

This section describes the Third-Party Handler (TPH) DLL.

TPH DLL Entry Points

GroupWise calls the TPH DLL Entry Points using their ordinal values. There are seven and each one must be exported even though not all seven are used. Please note the required ordinal value of each function.

This section contains information on the following entry point methods:

TPHVersion

Reserved. Even though it is currently not called by GroupWise, it must be included. This function must be exported as ordinal value 1 (one).

Syntax

```
DWORD WINAPI TPHVersion ( VOID )
```

Parameters

None.

Return Values

0.

Example

```
DWORD WINAPI TPHVersion ( void )
{
     return 0;
}
```

Compatibility

Called during the initialization process. It passes the GroupWise client version to the TPH via this method. You can determine the version by evaluating the high and low order bytes of the `AppVersion` parameter. This function must be exported as ordinal value 2 (two).

Syntax

```
DWORD WINAPI Compatibility (
   ATOM AppName,
   WORD AppVersion );
```

Parameters

AppName

Atom of the calling application. See a Windows programming manual for Atoms information.

AppVersion

GroupWise version being loaded for the calling application.

Return Values

Make sure the Compatibility() return value is TRUE or FALSE cast as a DWORD value. An incompatible DLL returns FALSE.

Example

```
DWORD WINAPI Compatibility( ATOM AppAtom, WORD AppVersion)
{
     // Greater than or equal to GroupWise 5.2
     if (5 >= LOBYTE(AppVersion) && 2 >= HIBYTE(AppVersion))
     {
          return (DWORD) TRUE;
     }
     else
     {
          return (DWORD) FALSE;
     }
}
```

Remarks

`Compatibility()` lets a DLL specify the GroupWise version to load it with and ensures compatibility between GroupWise and third-party DLLs. Your DLL can verify major and minor GroupWise versions, then return GroupWise a value indicating whether to load your DLL. A Novell application name and GroupWise version are passed as parameters to Compatibility(). You must export `Compatibility()` at ordinal value 2 (two).

The GroupWise AppName is GROUPWISE. The AppVersion is a word value with the low-order byte containing the GroupWise major version, and the high-order byte containing the minor version.

TimeStamp

Reserved. Even though it is currently not called by GroupWise, it must be included. This function must be exported as ordinal value 3 (three).

Syntax

```
DWORD WINAPI TimeStamp ( VOID )
```

Parameters

None

Return Values

0.

Example

```
DWORD WINAPI TimeStamp( void )
{
      return 0;
}
```

Entry

Entry() is called to allow the DLL to perform any required initialization. It is also called prior to any tokens being passed to the DLL. The DLL is provided with the resource language code to indicate the language being used in the calling application. This function must be exported at ordinal value 4 (four).

Syntax

```
WORD WINAPI Entry ( WORD wLanguage )
```

Parameters

wLanguage

The default language for the calling application. For the US version of GroupWise, the value of the wLanguage parameter would be:

LOBYTE ’U’
HIBYTE 'S'

Return Values

A value greater than zero should be returned to indicate success. A return value less than or equal to zero indicates an error occurred.

Example

```
WORD WINAPI Entry( WORD wLanguage)
{
     if ('U' == (char) LOBYTE(wLanguage) && 'S' == (char) HIBYTE(wLanguage))
     {
          return 1;
     }
     else
     {
          return 0;
     }
}
```

Exit

Allows the DLL to execute any code for cleanup purposes (closing files or destroying dialogs) before the DLL is unloaded. This function must be exported as ordinal value 5 (five).

Syntax

```
WORD WINAPI Exit ( VOID );
```

Parameters

None.

Return Values

The return value is ignored.

Example

```
WORD WINAPI Exit( void )
{
     return 1;
}
```

HandleToken

Called whenever a token is published by the GroupWise client. The token is then passed as a parameter.This is the heart of the TPH process. By evaluating the token parameter, you can determine what token was published as well as any token related data in the token structure. This function must be exported as ordinal value 6 (six).

Syntax

```
int WINAPI HandleToken (
   LPTPH_RETURNVAL lpTokenData
   HWND hLinkWnd,
   WORD msg );
```

Parameters

lpData

A LPTPH_RETURNVAL structure contains two members. The first member is a pointer to an LPMAC_TOKEN structure. This structure contains the token ID and parameter data. The second member is a pointer to an LPMAC_RETURNVAL structure. This structure is used for return values from value-returning tokens.

hLinkWnd

Handle to the link window.

msg

Link window execute message.

Return Values

The following values can be returned.

DLL_HAN_NOT_HANDLED	The token was not processed.
DLL_HAN_NO_ERROR	The token was processed. This value should be returned to block a token.
DLL_HAN_NOT_FOUND	The token resulted in a GroupWise "not found" condition.
DLL_HAN_CANCEL	The user cancelled the function.
DLL_HAN_TOKEN_ERROR	The token was not valid.
DLL_HAN_PARM_ERROR	One or more of the parameters were invalid.

Example

```
int WINAPI HandleToken(LPTPH_RETURNVAL lpTokenData, HWND hLinkWnd, WORD msg )
{
     return DLL_HAN_NOT_HANDLED;
}
```

Remarks

HandleToken( ) is called using ordinal value 6 (six).

ValidateToken

Reserved. Even though it is currently not called by GroupWise, it must be included. This function must be exported as ordinal value 7 (seven).

Syntax

```
DWORD WINAPI ValidateToken (
   LPMAC_TOKEN lpTokenData,
   LPVOID lpDataTypeInfo );
```

Parameters

Not used.

Return Values

0.

Example

```
DWORD WINAPI ValidateToken(LPMAC_TOKEN lpTokenData, Lpvoid lpDataTypeInfo)
{
   return DLL_VAL_UNKNOWN_TOKEN;
}
```

THP DLL Memory Allocation

The following methods are used in memory allocation:

- “AllocPtr” on page 51
- “FreePtr” on page 52
- “InitTkn” on page 53
- “AddTknParm” on page 54
- “SendTkn” on page 55

AllocPtr

Allocates memory and called by InitToken( ).

Definition

```
LPVOID WINAPI AllocPtr ( int nSize, WORD wFlags )
{
   LPVOID lpvPtr;

   if ((hMem = GlobalAlloc (wFlags, nSize)) != NULL )

   {
      if (( lpvPtr = GlobalLock (hMem)) != NULL )
      return lpvPtr;

      else{GlobalFree (hMem);
      return NULL;
      }
   }
   else
   return NULL;
}
```

FreePtr

Frees memory allocated by AllocPtr( ).

Definition

```
void WINAPI FreePtr ( LPVOID lpvPtr )
{
   HGLOBAL hMem;
   if (( hMem = (HGLOBAL)LOWORD(GlobalHandle( SELECTOROF(lpvPtr)))) != NULL)

   {
      if (!(GlobalUnlock (hMem)))
      GlobalFree (hMem);
   }
}
```

InitTkn

Attempts to allocate a memory block to contain a token and its specified number of parameters.

Definition

```
void WINAPI InitTkn ( WORD nParmCount )
{ //Allocate one less mac_param because 1 is already
   defined in theMAC_TOKEN structure
if ((lpTknData = (LPMAC_TOKEN)AllocPtr (sizeof (MAC_TOKEN)
   +(sizeof(MAC_PARAM) *(nParmCount-1) + 100), GHND)) ==
   NULL)
{
   lpTknData = (LPMAC_TOKEN)NULL;
   }
}
```

AddTknParm

Adds parameter values to a previously allocated MAC_PARAM structure. This token should be called once for each of the token's parameters. If a token parameter is optional, the token value should be set to 0x8000 to inform GroupWise to ignore the parameter. Otherwise it should be 00000.

Definition

```
void WINAPI AddTknParm ( MAC_VALUE_TYPE eType, WORD
   cParmNum, WORD wFlags, LPVOID lpvParm, DWORD dwParm,
   double dbParm )
{
   lpTknData-DataBlock[cParmNum].eType = eType;
   lpTknData-DataBlock[cParmNum].wFlags = wFlags;

   if (dwParm != 0)
      lpTknData-DataBlock[cParmNum].uData.dwValue = dwParm;
   else
  if (dbParm != 0)
      lpTknData-DataBlock[cParmNum].uData.drValue = dbParm;
   else
   if (lpvParm != NULL)
      lpTknData-DataBlock[cParmNum].uData.lpvPtr = lpvParm;
}
```

SendTkn

Once the token and parameter structures have been allocated and initialized, your DLL can send the token to GroupWise by sending a message to the link window.

Definition

```
DWORD WINAPI SendTkn ( MAC_TOKENID wTknID, WORD
   nParmCnt,LPLPMAC_RETURNVAL lplpTknReturn )
{
   TPH_RETURNVAL TknData;
   LPTPH_RETURNVAL lpTPHData;

   // Fill global token structure //
   lpTknData-hszCommand = NULL;
   lpTknData-hszRequestor = (HSZ)hRequestor;
   lpTknData-dwMacroID = 0L;
   lpTknData-wTokenID = wTknID;
   lpTknData-cParam = nParmCnt;
   lpTknData-dwFlags = NULL;
   lpTknData-dwReserved = NULL;

   // Fill ReturnVal structure //
   TknData.lpTokenData = lpTknData;
   TknData.lplpmacRetVal = lplpTknReturn;

   lpTPHData = TknData;

   // Valuereturning token //
   dwRV = SendMessage (WPLnkWnd,
     WPLM_EXECUTE_VALRETURN_TOKEN, 0,
     (LPARAM)lpTPHData);FreePtr (lpTknData);return dwRV;
} // End of SendTkn( ) function //
```

Return Values

The following values can be returned.

Return Value	Description
DLL_HAN_NOT_HANDLED	The token was not processed.
DLL_HAN_NO_ERROR	The token was processed with no error.
DLL_HAN_NOT_FOUND	The token resulted in a GroupWise not found condition.
DLL_HAN_CANCEL	The user cancelled the function.
DLL_HAN_TOKEN_ERROR	The token was not valid.
DLL_HAN_PARM_ERROR	One or more of the parameters were invalid.

Remarks

- lpTPHData is a long pointer to the TPH_RETURNVAL structure.
- WPLM_EXECUTE_VALRETURN_TOKEN is the *msg* parameter passed to TokenHandle.

- hLinkWindow is the link window handle passed to TokenHandle.
- SendMessage returns a value indicating the token processing result defined below. If the message is successful and the token can return a value, TPH_RETURNVAL structure (lpTPHData) is filled in with data.

SendTkn sends the token structure to the link window using SendMessage.

4 Token Commander API Reference

The GroupWise Execute() (page 58) Token Commander is an API that allows you to execute a token command.

Execute()

Allows you to execute a token command.

Syntax

```
WORD Execute(ANSISTRING CMD; ANSISTRING RetString;)
```

Parameters

`CMD As ANSISTRING`

Token command string to send to GroupWise (example 'AboutDlg()')

`RetString As ANSISTRING`

Empty string if token doesn't have a return value. Return string if token has a return value and execution succeeds. Error string if token execution fails.

Return Values

WORD. 0 if token execution fails. 1 if token execution succeeds.

Remarks

To get the currently logged in user ID, use the EnvUserID() token. The Pascal code to send this token is:

```
vCommander:=CreateOleObject('GroupWiseCommander');
Retval:=vCommander.Execute('EnvUserID',ReturnString)
```

ReturnString = the logged in GroupWise UserID string.

Retval = 1 for success

When calling the token commander execute method from .NET, you need to initialize the "RetString" output parameter with the correct type or you get a "Type Mismatch" exception from the .NET runtime. For example:

C#:

```
// Invalid: Produces "Type Mismatch"
string strResult;
commander.Execute("AboutDlg()", out strResult);

// Valid
string strResult = "";
commander.Execute("AboutDlg()", out strResult);
```

Visual Basic.NET:

```
// Invalid: Produces "Type Mismatch"
Dim strResult As String
commander.Execute("AboutDlg()", strResult)


// Valid
Dim strResult As String
strResult = “”
commander.Execute("AboutDlg()", strResult)
```

5 Tokens Reference (A-K)

Tokens are low-level events that are packaged into a meaningful representation and are subscribed or published by one of the Token functions (see “Subscribe and Publish” on page 22.)

Information on each token is contained in the following alphabetical sections:

For more token information, see Chapter 6, “Tokens Reference (L-Z),” on page 541.

A

This section contains information about the following tokens:

AboutDlg()

Displays the GroupWise About dialog box, which contains product, network, and post office information.

Token ID

```
DTKN_ABOUT or 66
```

Syntax

```
VOID AboutDlg()
```

AccountsDlg()

Launches the account options dialog that allows users to create, delete, and modify POP3, IMAP4, and GroupWise accounts.

Token ID

```
DTKN_ACCOUNTS or 130
```

Syntax

```
Void AccountsDlg()
```

AddNewChecklistItem()

Activates in-place creation of a new checklist item, if the current focus is in a checklist folder.

Token ID

```
BFTKN_CHECKLIST_NEW_ITEM 1102
```

Syntax

```
void AddNewChecklistItem()
```

AccountSync()

Synchronizes all the accounts marked in the account options dialog with their respective servers.

Token ID

```
BFTKN_SYNC_ACCOUNT or 285
```

Syntax

```
Void AccountSync([ANSISTRING AccountId])
```

Parameters

AccountId As ANSISTRING

(Optional) The Account ID.

AddNewDocument()

Creates a new document in the GroupWise library.

Token ID

```
AFTKN_DM_NEW_DOC or 858
```

Syntax

```
ANSISTRING AddNewDocument([ANSISTRING Extension])
```

Parameters

Extension As ANSISTRING

(Optional) Text string to specify extension as seen on GW property sheet. String specifier for File extension field listed as a property of the document version.

Return Values

DocIDStr As ANSISTRING

DocumentID. Returns the Library, Document Number, and version in the string format Domain.PostOffice.Library.Doc#.Ver#.

See Also

Documents.Add method in the Object API documentation (http://developer.novell.com/ndk/gwobjapi.htm).

AddressBookCompare()

Returns a Boolean value indicating whether two addresses point to the same person or resource.

Token ID

```
AFTKN_AB_COMPARE or 774
```

Syntax

```
BOOLEAN AddressBookCompare(ANSISTRING Address1;
                           ANSISTRING Address2)
```

Parameters

Address1 As ANSISTRING

First UserID or EmailAddress in DPU format (Domain.PostOffice.UserID)

Address2 As ANSISTRING

Second UserID or EmailAddress in DPU format (Domain.PostOffice.UserID)

Return Values

BOOLEAN. True, if two address strings point to the same person or resource. If not, it returns False.

See Also

“AddressBookResolve()” on page 76

AddressBookDlg()

Displays the dialog box in the Address Book which contains user names and IDs, groups, and resources.

Versions: GroupWise 5.2 and 5.5 only

Token ID

```
DTKN_ADDRBOOK or 44
```

Syntax

```
VOID AddressBookDlg()
```

AddressBookGetEntry()

Returns the contents of an Address Book field.

Token ID

```
AFTKN_AB_GET_ENTRY or 718
```

Syntax

```
ANSISTRING AddressBookGetEntry (ENUM ABField;
                                ANSISTRING FullAddressText;
                                ANSISTRING [UserID])
```

Parameters

ABField As ENUM

Address Book Field. The entry returns in a variable. If a field does not exist in a specified address section, a run-time error occurs. Fields include:

199	PersonalGroupsID!
202	PublicGroupsDescription!
203	PublicGroupsDomain!
204	PublicGroupsHost!
205	PublicGroupsID!
208	ResourcesDescription!
209	ResourcesDomain!
210	ResourcesFID!
211	ResourcesHost!
212	ResourcesID!
213	ResourcesOwner!
214	ResourcesType!
244	UsersAccountID!
245	UsersDept!
246	UsersDomain!
247	UsersFID!
248	UsersFaxNum!
249	UsersFirstName!
250	UsersHost!
251	UsersID!

252	UsersLastName!
253	UsersNetID!
254	UsersPhone!
255	UsersTitle!
256	UsersUD1!
	UsersUD1! through UsersUD9! are defined by the system administrator.
257	UsersUD10!
258	UsersUD2!
259	UsersUD3!
260	UsersUD4!
261	UsersUD5!
262	UsersUD6!
263	UsersUD7!
264	UsersUD8!
265	UsersUD9!

FullAddressText As ANSISTRING

The full EmailAddress string of a person or resource. The command fails if the FullAddressText is not unique. Use AddressBookResolve to determine if the FullAddressText is unique.

UserID As ANSISTRING

(Optional) User ID of the mailbox that contains the Address Book information. Use this parameter to search a proxy mailbox.

Return Values

FieldEntry As ANSISTRING. For example, UsersFirstName! may return Ken or Tammy.

AddressBookGetField()

Returns the name of an Address Book field.

Token ID

```
AFTKN_AB_GET_FIELD or 719
```

Syntax

```
ANSISTRING AddressBookGetField (ENUM ABField)
```

Parameters

ABField As ENUM

Address Book Field. The entry returns in a variable. If a field does not exist in a specified address section, a run-time error occurs. Fields include the following:

199	PersonalGroupsID!
202	PublicGroupsDescription!
203	PublicGroupsDomain!
204	PublicGroupsHost!
205	PublicGroupsID!
208	ResourcesDescription!
209	ResourcesDomain!
210	ResourcesFID!
211	ResourcesHost!
212	ResourcesID!
213	ResourcesOwner!
214	ResourcesType!
244	UsersAccountID!
245	UsersDept!
246	UsersDomain!
247	UsersFID!
248	UsersFaxNum!
249	UsersFirstName!
250	UsersHost!
251	UsersID!
252	UsersLastName!
253	UsersNetID!

254	UsersPhone!
255	UsersTitle!
256	UsersUD1!
	UsersUD1! through UsersUD9! are defined by the system administrator.
257	UsersUD10!
258	UsersUD2!
259	UsersUD3!
260	UsersUD4!
261	UsersUD5!
262	UsersUD6!
263	UsersUD7!
264	UsersUD8!
265	UsersUD9!

Return Values

FieldName As ANSISTRING

For example, PublicGroupsDomain! returns the value "Domain."

AddressBookGetFullName()

Returns the full name associated with a user ID in an Address Book.

Token ID

```
AFTKN_AB_GET_FULLNAME or 818
```

Syntax

```
ANSISTRING AddressBookGetFullName (ANSISTRING
                                   FullAddressText;
                                   [ANSISTRING UserID])
```

Parameters

FullAddressText As ANSISTRING

EmailAddress in the form "Domain.PostOffice.UserID." Will also except UserID as well as EmailAddress.

UserID As ANSISTRING

(Optional) User ID of the mailbox that contains the Address Book information. Use this parameter to search a proxy mailbox.

Return Values

FullName As ANSISTRING

The descriptive name or displayName displayed to the user.

AddressBookResolve()

Returns a list of valid, unambiguous (unique) addresses.

Token ID

```
AFTKN_AB_RESOLVE or 794
```

Syntax

```
ANSISTRING AddressBookResolve(ANSISTRING Addresses;
                              ENUM RemoveAmbiguousAddresses;
                              ENUM RemoveInvalidAddresses;
                              [ANSISTRING UserID])
```

Parameters

Addresses As ANSISTRING

Separate addresses with commas and enclose in double quotation marks.

RemoveAmbiguousAddresses As ENUM

Specifies what to do with ambiguous addresses as follows:

160	Fail! Causes the command to fail if there are ambiguous addresses.
201	Prompt! Displays a dialog box where the user selects an address to include.
1	Yes! Removes all ambiguous addresses.

RemoveInvalidAddresses As ENUM

Specifies what to do with invalid addresses as follows:

160	Fail! Causes the command to fail if there are invalid addresses.
201	Prompt! Displays a dialog box where the user removes invalid addresses from the address list.
1	Yes! Removes all invalid addresses.

UserID As ANSISTRING

(Optional) User ID of the mailbox that contains the Address Book information. Use this information to search in a proxy mailbox.

Return Values

EmailAddress(es) As ANSISTRING

FullAddressText Domain.PostOffice.UserID of valid and unique addresses.

AddressBookResolveFullName()

Returns the full email address of a name or resource.

Token ID

```
AFTKN_AB_RESOLVE_FULLNAME or 817
```

Syntax

```
ANSISTRING AddressBookResolveFullName (ANSISTRING FullName;
                                       [ANSISTRING UserID])
```

Parameters

FullName As ANSISTRING

User or resource’s DisplayName, such as "First Last" or UserID.

UserID As ANSISTRING

(Optional) User ID of the mailbox that contains the Address Book information. Use this information to search in a proxy mailbox.

Return Values

EmailAddress As ANSISTRING

Full EmailAddress or FullAddressText (Domain.PostOffice.UserID).

AddressItemDlg()

Causes the Address Book dialog to be displayed. This token is only valid when a Compose view is open and active. The addresses selected in the Address Book dialog will be set in the active Compose view.

Token ID

```
DTKN_ADDRBOOK_MODAL or 108
```

Syntax

```
VOID AddressItemDlg()
```

AddressListAdd()

Adds a user ID to To, CC, and BC address Lists. A run-time error occurs for incorrect or non-existent user IDs.

Token ID

```
AFTKN_AL_ADD or 751
```

Syntax

```
VOID AddressListAdd(DWORD Handle;
                    [ANSISTRING ToList];
                    [ANSISTRING CCList];
                    [ANSISTRING BCList])
```

Parameters

Handle As DWORD

Address List handle, returned by AddressListCreate() or AddressListCreateFromGroup.

ToList As ANSISTRING

(Optional) User ID.

CCList As ANSISTRING

(Optional) User ID.

BCList As ANSISTRING

(Optional) User ID.

AddressListCreate()

Creates an address list.

Token ID

```
AFTKN_AL_CREATE or 689
```

Syntax

```
DWORD AddressListCreate([ANSISTRING UserID])
```

Parameters

UserID As ANSISTRING

(Optional) User ID of the mailbox where the address list is being built. Use this parameter to create an address list in a proxy mailbox.

Return Values

Handle As DWORD

Address list handle, or 0 if an error occurs.

AddressListCreateFromGroup()

Constructs an address list from a group or distribution list. Creates an address list from a public or personal group. Default: personal group. To specify a public group, use the full address string. Not recordable.

Token ID

```
AFTKN_AL_CREATE_FROM_GROUP or 740
```

Syntax

```
DWORD AddressListCreateFromGroup(ANSISTRING GroupName;
                                 [ANSISTRING UserID])
```

Parameters

GroupName As ANSISTRING

Public or personal group name, such as the name of a distribution list. If domain and host are absent, the GroupName is assumed to be personal.

UserID As ANSISTRING

(Optional) User ID of the mailbox that contains the Address Book information. Use this parameter to search a proxy mailbox.

Return Values

DWORD. Handle to the list of addresses in the group. Check the return value for a value of zero, which means an error occurred.

Remarks

IMPORTANT: Delete the address list handle when it is no longer needed, or system resources will be lost. See AddressListDelete() (page 83).

For personal groups, ensure that the personal address book in which the group resides is listed as a selected book in the Name Completion Search Order dialog in the GroupWise client. Otherwise, the token fails.

See Also

AddressListDelete() (page 83)

AddressListDelete()

Deletes an address list handle from memory.

Token ID

```
AFTKN_AL_DELETE or 690
```

Syntax

```
VOID AddressListDelete(DWORD Handle)
```

Parameters

Handle As DWORD

Address list handle.

See Also

“AddressListCreate()” on page 81

AddressListEdit()

Displays a dialog box for editing an address list. The Address Book dialog box is initialized with the contents of the address list. Changes to the TO, CC, and BC dialog lists replace the corresponding contents of the address list.

Token ID

```
AFTKN_AL_EDIT or 752
```

Syntax

```
VOID AddressListEdit(DWORD Handle)
```

Parameters

Handle As DWORD

Address list handle, returned by AddressListCreate().

See Also

“AddressBookDlg()” on page 70
“AddressListDelete()” on page 83

AddressListGetAddress()

Returns the full string of a specified address.

Token ID

```
AFTKN_AL_GET_ADDRESS or 691
```

Syntax

```
ANSISTRING AddressListGetAddress(DWORD Handle;
                                 WORD Index;
                                 ENUM FromWhichList)
```

Parameters

Handle As DWORD

Address list handle, returned by AddressListCreate().

Index As WORD

Address index. The first address is 0, the second is 1, and so forth. If the index is out of range, a run-time error occurs. See “AddressListGetCount()” on page 86.

FromWhichList As ENUM

Address list to apply index to:

124	BCList!
132	CCList!
137	Combined!
	Concatenates TO, CC, and BC, respectively.
233	ToList!

Return Values

EmailAddress As ANSISTRING

See Also

“AddressListDelete()” on page 83
“AddressListGetAddress()” on page 85

AddressListGetCount()

Returns the number of addresses in a list.

Token ID

```
AFTKN_AL_GET_COUNT or 692
```

Syntax

```
WORD AddressListGetCount(DWORD Handle;
                         ENUM FromWhichList)
```

Parameters

Handle As DWORD

Address list handle, returned by AddressListCreate().

FromWhichList As ENUM

Address list to apply index to:

124	BCList!
132	CCList!
137	Combined!
	Concatenates TO, CC, and BC, respectively.
233	ToList!

Return Values

Count As WORD

Number of addresses.

AddressListGetEntry()

Returns the contents of an address list entry.

Versions: GroupWise 5.5 only

Token ID

```
AFTKN_AL_GET_ENTRY or 693
```

Syntax

```
ANSISTRING AddressListGetEntry(DWORD Handle;
                               WORD Index;
                               ENUM ABField;
                               ENUM FromWhichList)
```

Parameters

Handle As DWORD

Address list handle, returned by AddressListCreate().

Index As WORD

Address index. The first address is 0, the second is 1, and so forth. If the index is out of range, a run-time error occurs.

ABField As ENUM

Address Book field. The field's entry returns in a variable. If a field does not exist in a section of the specified address, a run-time error occurs. Fields include:

199	PersonalGroupsID!
202	PublicGroupsDescription!
203	PublicGroupsDomain!
204	PublicGroupsHost!
205	PublicGroupsID!
208	ResourcesDescription!
209	ResourcesDomain!
210	ResourcesFID!
211	ResourcesHost!
212	ResourcesID!
213	ResourcesOwner!
214	ResourcesType!
244	UsersAccountID!
245	UsersDept!

246	UsersDomain!
247	UsersFID!
248	UsersFaxNum!
249	UsersFirstName!
250	UsersHost!
251	UsersID!
252	UsersLastName!
253	UsersNetID!
254	UsersPhone!
255	UsersTitle!
256	UsersUD1!
	UsersUD1! through UsersUD9! are defined by the system administrator.
257	UsersUD10!
258	UsersUD2!
259	UsersUD3!
260	UsersUD4!
261	UsersUD5!
262	UsersUD6!
263	UsersUD7!
264	UsersUD8!
265	UsersUD9!

FromWhichList As ENUM

Address list to apply index to:

124	BCList!
132	CCList!
137	Combined!
	Concatenates TO, CC, and BC, respectively.
233	ToList!

Return Values

FieldValue As ANSISTRING

Number of addresses.

See Also

“AddressListDelete()” on page 83

AddressListGetField()

Returns an address list field name.

Versions: GroupWise 5.5 only

Token ID

```
AFTKN_AL_GET_FIELD or 694
```

Syntax

```
ANSISTRING AddressListGetField(DWORD Handle;
                               WORD Index;
                               ENUM ABField;
                               ENUM FromWhichList)
```

Parameters

Handle As DWORD

Address list handle, returned by AddressListCreate().

Index As WORD

Address index. The first address is 0, the second is 1, and so forth. If the index is out of range, a run-time error occurs.

ABField As ENUM

Address Book field. The field's entry returns in a variable. If a field does not exist in a section of the specified address, a run-time error occurs. Fields include:

199	PersonalGroupsID!
202	PublicGroupsDescription!
203	PublicGroupsDomain!
204	PublicGroupsHost!
205	PublicGroupsID!
208	ResourcesDescription!
209	ResourcesDomain!
210	ResourcesFID!
211	ResourcesHost!
212	ResourcesID!
213	ResourcesOwner!
214	ResourcesType!
244	UsersAccountID!
245	UsersDept!

246	UsersDomain!
247	UsersFID!
248	UsersFaxNum!
249	UsersFirstName!
250	UsersHost!
251	UsersID!
252	UsersLastName!
253	UsersNetID!
254	UsersPhone!
255	UsersTitle!
256	UsersUD1!
	UsersUD1! through UsersUD9! are defined by the system administrator.
257	UsersUD10!
258	UsersUD2!
259	UsersUD3!
260	UsersUD4!
261	UsersUD5!
262	UsersUD6!
263	UsersUD7!
264	UsersUD8!
265	UsersUD9!

FromWhichList As ENUM

Address list to apply index to:

124	BCList!
132	CCList!
137	Combined!
	Concatenates TO, CC, and BC, respectively.
233	ToList!

Return Values

FieldName As ANSISTRING. Returns an Address Book field name.

See Also

“AddressListDelete()” on page 83

AddressListGetFullName()

Returns the full name associated with a user ID in an address list.

Versions: GroupWise 5.5 only

Token ID

```
AFTKN_AL_GET_FULLNAME or 819
```

Syntax

```
ANSISTRING AddressListGetFullName(DWORD Handle;
                                  WORD Index;
                                  ENUM FromWhichList)
```

Parameters

Handle As DWORD

Address list handle, returned by AddressListCreate().

Index As WORD

Address index. The first address is 0, the second is 1, and so forth. If the index is out of range, a run-time error occurs.

FromWhichList As ENUM

Address list to apply index to:

124	BCList!
132	CCList!
137	Combined!
	Concatenates TO, CC, and BC, respectively.
233	ToList!

Return Values

DisplayName As ANSISTRING. Full name.

See Also

“AddressListDelete()” on page 83

AddressListIsInSection()

Lets you determine if an address already exists in a specified address list. Returns the number of addresses in a section (such as To, CC, BC lists), of a list. Not recordable.

Token ID

```
AFTKN_AL_IS_IN_SECTION or 710
```

Syntax

```
BOOLEAN AddressListIsInSection(DWORD Handle;
                               WORD Index;
                               ENUM Section;
                               ENUM FromWhichList)
```

Parameters

Handle As DWORD

Address list handle, returned by AddressListCreate or AddressListCreateFromGroup.

Index As WORD

Address index. The first address is 0, the second is 1, and so forth. If the index is out of range, a run-time error occurs.

Section As ENUM

The section to search in, as follows:

159	ExternalAddess!
3	PersonalGroups!
2	PublicGroups!
1	Resources!
0	Users!

FromWhichList As ENUM

Address list to apply Index to:

124	BCList!
132	CCList!
137	Combined!
	Concatenates TO, CC, and BC, respectively.
233	ToList!

Return Values

BOOLEAN. If the address found at AddressIndex is in section, then return True, otherwise return False.

Remarks

If `FromWhichList` is combined or not provided, then use the address found at `Index`.The Combined list is a combination of To, CC, and BC addresses (in that order).

Index is zero-based, meaning the first address in the list has an index of 0 and the last item has an index of AddressListGetCount(Combined)1.

For example if AddressListGetCount is equal to an index of 10 items, the first item index is 0 and last item index is 9. In this example, if the address is between 0 and 9 then return True, otherwise if it is out of that range, return False. Possible scenarios include:

Combined or not provided (Combined includes To, CC, and BC)

Last Item = AddressListGetCount(Combined) - 1

To List

Last Item = AddressListGetCount(ToList) - 1

CcList

Last Item = AddressListGetCount(CcList) - 1

BcList

Last Item = AddressListGetCount(BcList) - 1

See Also

AddressListCreate() (page 81)
AddressListCreateFromGroup() (page 82)
AddressListGetCount() (page 86)

AlarmSet()

Sets an alarm that notifies a user of an appointment and optionally launches a program. The appointment must have the input focus and must be future.

Token ID

```
AFTKN_SET_ALARM or 595
```

Syntax

```
VOID AlarmSet(WORD HoursBefore;
              WORD MinutesBefore
              [ANSISTRING ProgramToLaunch])
```

Parameters

HoursBefore As WORD

Number of hours before the appointment to display the alarm.

MinutesBefore As WORD

Number of minutes before the appointment to display the alarm.

ProgramToLaunch As ANSISTRING

(Optional)

Remarks

This token requires an Appointment Item view to be active.

AlarmSetDlg()

Displays the Set Alarm dialog box. The appointment must have the input focus and must be future.

Token ID

```
DTKN_SET_ALARM or 84
```

Syntax

```
VOID AlarmSetDlg()
```

AlternateTimeZone()

Shows or hides additional timezone on the calendar.

Token ID

```
AFTKN_ALTERNATE_TIMEZONE or 993
```

Syntax

```
VOID AlternateTimeZone(
      [BOOL] DisplayAlternateTimeZone;
      [WIDESTRING] AlternateTZRegKey)
```

Parameters

DisplayAlternateTimeZone As BOOL

(Optional) TRUE to display the alternate time zone, FALSE to hide it.

AlternateTZRegKey As WIDESTRING

(Optional) Identifies one of the subkey's under "HKEY_LOCAL_MACHINE\Software\Microsoft\Windows NT\Time Zones" to use as the new alternate time zone. For instance, "Alaskan Standard Time".

AlternateTimeZoneDlg()

Displays a dialog enabling the user to choose to display an additional time zone on the calendar.

Token ID

```
DTKN_ALTERNATE_TIMEZONE or 143
```

Syntax

```
VOID AlternateTimeZoneDlg()
```

AppAllowClose()

Specifies whether a user can exit GroupWise. This command is typically called by a DDE client or cross-application macro. GroupWise cannot be closed if a macro is playing or if AppAllowClose is set to No! (See Allow parameter).

Token ID

`AFTKN_ALLOW_EXIT or 775`

Syntax

```
VOID AppAllowClose([ENUM Allow])
```

Parameters

Allow As ENUM

(Optional) If Allow is set to No!, you must set it to Yes! before ending the DDE conversation or cross-application macro. Otherwise, you will be able to exit GroupWise only through the Windows Task List.

0 No
1 Yes

Remarks

AppClose() causes GroupWise to exit, unless AppAllowClose(No!) has previously been called. If you call AppAllowClose(Yes!), followed by AppClose(), the client should exit.

AppClose()

Causes the GroupWise client to shut down. AppAllowClose must be set to Yes! for AppClose to function.

Token ID

`BFTKN_APP_EXIT or 160`

Syntax

```
VOID AppClose()
```

Remarks

AppClose() causes GroupWise to exit, unless AppAllowClose(No!) has previously been called. If you call AppAllowClose(Yes!), followed by AppClose(), the client should exit.

AppointmentSetInterval()

Sets the time interval on the appointments part of the calendar view.

Token ID

```
BFTKN_SET_APPT_DISP_INTVL or 489
```

Syntax

```
VOID AppointmentSetInterval([WORD Interval])
```

Parameters

Interval As WORD

(Optional) Valid intervals are: 10, 15, 20, 30, 60, 120.

AppointmentSetMode()

Changes the display mode for the calendar appointment view.

Token ID

`BFTKN_SET_APPT_DISP_MODE or 488`

Syntax

```
VOID AppointmentSetMode([ENUM Mode])
```

Parameters

Mode As ENUM

(Optional) As follows:

0	Grid! Default if no parameter given.
1	Text!

AttachmentAdd()

Attaches a file to an item.

Token ID

```
AFTKN_ADD_ATTACHMENT or 572
```

Syntax

```
VOID AttachmentAdd(ANSISTRING Filename;
                   [ENUM DeleteWhenDone];
                   [ANSISTRING AttachmentDisplayName])
```

Parameters

Filename As ANSISTRING

DeleteWhenDone As ENUM

(Optional)

0 No
1 Yes

AttachmentDisplayName As ANSISTRING

(Optional) Name displayed in the Attach List box (up to 12 and 32 characters)

AttachmentAddDlg()

Displays the AddAttachment dialog box to attach a file to an item.

Token ID

```
DTKN_ATTACH_FILE or 88
```

Syntax

```
VOID AttachmentAddDlg([ANSISTRING DirectoryPath])
```

Parameters

DirectoryPath As ANSISTRING

(Optional)

See Also

“AttachmentAdd()” on page 103
“AttachmentListDlg()” on page 107

AttachmentDelete()

Deletes an attachment before an item is sent.

Token ID

```
BFTKN_DELETE_ATTACH or 323
```

Syntax

```
VOID AttachmentDelete([WORD AttachmentIndex])
```

Parameters

AttachmentIndex WORD

(Optional) Index number (zero-based).

AttachmentDocReference()

Displays the user interface to allow a user to select a document reference that will be added as an attachment to the active item Compose view.

Token ID

```
DTKN_ATTACH_DOCREF or 95
```

Syntax

```
VOID AttachmentDocReference()
```

AttachmentListDlg()

Displays the Attachments dialog box to add attachments to a new item.

Token ID

```
DTKN_ATTACHMENTS or 81
```

Syntax

```
VOID AttachmentListDlg()
```

See Also

“AttachmentAdd()” on page 103
“AttachmentAddDlg()” on page 104
“AttachmentSaveAs()” on page 109

AttachmentOpen()

Opens an attachment.

Token ID

```
BFTKN_OPEN_ATTACH or 267
```

Syntax

```
VOID AttachmentOpen([WORD AttachmentIndex])
```

Parameters

AttachmentIndex As WORD

(Optional) Index number (zero-based).

See Also

“AttachmentAdd()” on page 103
“AttachmentAddDlg()” on page 104
“AttachmentSaveAs()” on page 109

AttachmentSaveAs()

Saves an attachment with a new file name. For this token to function, the message window with the attachment in the list must be showing.

Token ID

```
AFTKN_SAVE_AS or 591
```

Syntax

```
VOID AttachmentSaveAs(ANSISTRING NewFilename;
                      WORD AttachmentIndex)
```

Parameters

NewFilename As ANSISTRING

Name to save the file as.

AttachmentIndex as WORD

AttachmentSaveAsDlg()

Displays the Save As dialog box to save an attachment with a new file name.

Token ID

```
DTKN_VIEWER_SAVE_AS or 45
```

Syntax

```
VOID AttachmentSaveAsDlg([WORD AttachmentIndex])
```

Parameters

AttachmentIndex as WORD

(Optional) The index of the attachment you want to save.

AttachmentView()

Displays an attachment.

Token ID

```
BFTKN_VIEW_ATTACH or 360
```

Syntax

```
VOID AttachmentView([ENUM TypeToView];
                    [WORD AttachmentIndex])
```

Parameters

TypeToView As ENUM

(Optional) Specifies the type to view as follows:

1	Attachment!
10	Message!
0	zOldMessage!

AttachmentIndex As WORD

(Optional) Index of the desired attachment.

AttachmentViewSame()

Displays the specified attachment in the same viewer window as the current message (without opening a new viewer window).

Token ID

```
BFTKN_VIEW_ATTACH_SAME 1154
```

Syntax

```
void AttachmentViewSame([ENUM TypeToView];
                        [WORD AttachmentIndex])
```

Parameters

TypeToView As ENUM

(Optional) Specifies the type to view as follows:

1 Attachment
10 Message
0 zOldMessage

AttachmentIndex As WORD

(Optional) Index of the desired attachment.

AutoPilotToggle()

Turns on or off the auto pilot that syncs all marked accounts.

Token ID

```
BFTKN_TOGGLE_AUTO_PILOT or 999
```

Syntax

```
VOID AutoPilotToggle()
```

AutoSizeAllDayEventPane()

Automatically resizes the all-day-event pane.

Token ID

```
DTKN_CAL_BKG_COLOR_DLG or 1233
```

Syntax

```
VOID AutoSizeAllDayEventPane([ENUM AutoSize])
```

Parameters

AutoSize As ENUM

(Optional) Specifies whether to resize the all-day-event pane:

0 No
1 Yes

B-C

This section contains information about the following tokens:

BusySearch()

Searches one or more user's schedules and displays the result in the Choose Appointment Time dialog box. Enclose multiple IDs in double quotation marks, separated by columns. (see TO, CC, and BC parameters).

Token ID

```
AFTKN_BUSYSRCH or 635
```

Syntax

```
VOID BusySearch([ANSISTRING TO];
                [ANSISTRING CC];
                [ANSISTRING BC];
                [WORD StartDay];
                [WORD StartMonth];
                [WORD StartYear];
                [WORD StartMinute];
                [WORD StartHour];
                [WORD DurationMinutes];
                [WORD DurationHours];
                [ENUM AppointmentInformation];
                [WORD NumberOfDays];
                [ENUM Sunday];
                [ENUM Monday];
                [ENUM Tuesday];
                [ENUM Wednesday];
                [ENUM Thursday];
                [ENUM Friday];
                [ENUM Saturday];
                [WORD DisplayBeginMinute];
                [WORD DisplayBeginHour];
                [WORD DisplayEndMinute];
                [WORD DisplayEndHour])
```

Parameters

TO As ANSISTRING

(Optional) User IDs to include in search.

CC As ANSISTRING

(Optional) User IDs to receive a carbon copy.

BC As ANSISTRING

(Optional) User IDs to receive a blind copy.

StartDay As WORD

(Optional) Day on which to start searching.

StartMonth As WORD

(Optional) Month in which to start searching.

StartYear As WORD

(Optional) Year in which to start searching.

StartMinute As WORD

(Optional) Minute in which to start searching.

StartHour As WORD

(Optional) Hour in which to start searching.

DurationMinutes As WORD

(Optional)

DurationHours As WORD

(Optional)

AppointmentInformation As ENUM

(Optional)

281 Hide
282 Show

Number of Days As WORD

(Optional) Number of days to search.

Sunday As ENUM

(Optional)

0 No
1 Yes

Monday As ENUM

(Optional)

0 No
1 Yes

Tuesday As ENUM

(Optional)

0 No
1 Yes

Wednesday As ENUM

(Optional)

0 No
1 Yes

Thursday As ENUM

(Optional)

0 No
1 Yes

Friday As ENUM

(Optional)

0 No
1 Yes

Saturday As ENUM

(Optional)

0 No
1 Yes

DisplayBeginMinute As WORD

(Optional) Minute to start displaying search results.

DisplayBeginHour As WORD

(Optional) Hour to start displaying search results.

DisplayEndMinute As WORD

(Optional) Minute to stop displaying search results.

DisplayEndHour As WORD

(Optional) Hour to stop displaying search results.

BusySearchDlg()

Displays the Busy Search Settings dialog box to search one or more of the user's schedules.

Token ID

```
DTKN_BUSYSRCH or 82
```

Syntax

```
VOID BusySearchDlg()
```

ButtonBarReset()

Resets the specified toolbar to the shipping default.

Token ID

```
BFTKN_BTNBAR_RESET 1112
```

Syntax

```
void ButtonBarReset([ENUM Toolbar])
```

Parameters

Toolbar As ENUM

(Optional) Specifies the type as follows:

0 Main
2 ItemContext
3 FolderContext

ButtonBarSetStyle()

Specifies a GroupWise ToolBar style.

Token ID

```
AFTKN_BTNBAR_SETSTYLE or 736
```

Syntax

```
VOID ButtonBarSetStyle([ENUM Style];
                       [ENUM Wrap])
```

Parameters

Style As ENUM

(Optional) Specifies the style as follows:

0	PictureOnly, Single Row!
2	PictureAndText, Multiple Rows!

Wrap As ENUM

(Optional)

0 Off!
1 On!

ButtonBarShow()

Turns the Toolbar on or off.

Token ID

```
BFTKN_BTNBAR_SHOW or 176
```

Syntax

```
VOID ButtonBarShow([ENUM State])
```

Parameters

State As ENUM

(Optional)

0 Off!
1 On!

CachingRefresh()

Requests all items, rules, and address books (including the system address book) from the master mailbox.

Token ID

```
BFTKN_CACHE_REPRIME or 1052
```

Syntax

```
VOID CachingRefresh()
```

CachingRetrieveProperties()

Requests the latest copy of a sent item's properties from the master mailbox.

Token ID

```
BFTKN_CACHE_RETRIEVE_PROPERTIES or 1058
```

Syntax

```
VOID CachingRetrieveProperties()
```

CalendarBackgroundColorDlg()

Displays the Calendar Background Color dialog box.

Token ID

```
DTKN_CAL_BKG_COLOR_DLG or 141
```

Syntax

```
VOID CalendarBackgroundColorDlg()
```

CalendarCreateDlg()

Displays the Create New Calendar dialog box.

Token ID

```
DTKN_CREATE_CALENDAR or 139
```

Syntax

```
VOID CalendarCreateDlg()
```

CalendarHideIcons()

Hides the icons on the calendar view. This token does not work for the month view.

Token ID

```
BFTKN_CAL_HIDE_ICONS or 977
```

Syntax

```
VOID CalendarHideIcons()
```

CalendarPublish()

Configure a calendar for publishing to the internet.

Token ID

```
AFTKIN_CALENDAR_PUBLISH or 935
```

Syntax

```
VOID CalendarPublish(
     [WIDESTRING] FolderName;
     [ENUM] Publish;
     [ENUM] Mode;
     [DWORD] RelativeStartDays;
     [DWORD] RelativeEndDays;
     [ENUM] IncludePrivate;
     [ENUM] IncludeAttachments)
```

Parameters

FolderName As WIDESTRING

(Optional) The full path of the calendar folder to publish.

Publish As ENUM

(Optional)

0 No. Revoke access to a previously published calendar.
1 Yes. Publish calendar information to the internet.

Mode As ENUM

(Optional)

0 Entire. Publish all events on the calendar. RelativeStartDays and RelativeEndDays are ignored.
1 Relative. RelativeStartDays and RelativeEndDays specify a floating window of events that will be published. The list of events is dynamically updated every time a client requests that published calendar to include events that fall within a specified time period.
2 Absolute. RelativeStartDays and RelativeEndDays specify a fixed time period relative to today. The list of events are static and will not be updated as time goes on.

RelativeStartDays As DWORD

(Optional) The number of days forward or backward from the current day to start publishing events.

RelativeEndDays As DWORD

(Optional) The number of days forward or backward from the current day to stop publishing.

IncludePrivate As ENUM

(Optional)

0 No. Do not publish calendar events marked as private.
1 Yes. Publish calendar events even if they are marked private.

IncludeAttachments As ENUM

(Optional)

0 No. Do not publish information about attachments (such as documents and files) that are associated with a calendar event.

1 Yes. Publish information about attachments (such as documents and files) that are associated with the calendar events.

Remarks

Note that this token is only enabled when the GroupWise system is configured for calendar publishing by the administrator.

Note that a user’s primary calendar cannot be published to the Internet.

CalendarPublishPropDlg()

Display a dialog enabling the user to configure calendar publishing.

Token ID

`DTKN_CALENDAR_PUBLISH_PROP_DLG or 144`

Syntax

```
VOID CalendarPublishPropDlg()
```

Remarks

Note that this token is only enabled when the user is currently viewing a calendar that is eligible for publishing. The calendar must not be the user's main calendar. Additionally, the GroupWise system must be configured by the administrator to allow calendar publishing.

CalendarSendDlg()

Display a dialog allowing the user to send a copy of a calendar through an email.

Token ID

```
DTKN_CALENDAR_SEND_DLG or 150
```

Syntax

```
VOID CalendarSendDlg()
```

Remarks

Note that this token is only enabled when the user is currently viewing a calendar folder.

CalendarIsShowingAppts()

Returns TRUE if the calendar is showing appointments.

Token ID

```
BFTKN_CALENDAR_IS_SHOWING_APPTS 1091
```

Syntax

```
BOOLEAN CalendarIsShowingAppts()
```

Return Values

Returns TRUE if the calendar is showing appointments, FALSE otherwise.

Remarks

The calendar must have focus for this token to work properly.

CalendarIsShowingNotes()

Returns TRUE if the calendar is showing notes.

Token ID

```
BFTKN_CALENDAR_IS_SHOWING_NOTES 1093
```

Syntax

```
BOOLEAN CalendarIsShowingNotes()
```

Return Values

Returns TRUE if the calendar is showing notes, FALSE otherwise.

Remarks

The calendar must have focus for this token to work properly.

CalendarIsShowingTasks()

Returns TRUE if the calendar is showing tasks.

Token ID

```
BFTKN_CALENDAR_IS_SHOWING_TASKS 1092
```

Syntax

```
BOOLEAN CalendarIsShowingTasks()
```

Return Values

Returns TRUE if the calendar is showing tasks, FALSE otherwise.

Remarks

The calendar must have focus for this token to work properly.

CalendarShowAppts()

Shows or hides appointments in the calendar.

Token ID

```
BFTKN_CALENDAR_SHOW_APPTS 1088
```

Syntax

```
void CalendarShowAppts(BOOLEAN Show)
```

Parameters

Show as BOOLEAN

Specifies the type as follows:

TRUE Show appointments
FALSE Hide appointments

Remarks

The calendar must have focus for this token to work properly.

CalendarShowIcons()

Shows the icons on the calendar view. This token does not work for the month view.

Token ID

```
BFTKN_CAL_SHOW_ICONS or 976
```

Syntax

```
VOID CalendarShowIcons()
```

CalendarShowNotes()

Shows or hides notes in the calendar.

Token ID

```
BFTKN_CALENDAR_SHOW_NOTES 1090
```

Syntax

```
void CalendarShowNotes(BOOLEAN Show)
```

Parameters

Show as BOOLEAN

Specifies the type as follows:

TRUE Show notes
FALSE Hide notes

Remarks

The calendar must have focus for this token to work properly.

CalendarShowTasks()

Shows or hides tasks in the calendar.

Token ID

```
BFTKN_CALENDAR_SHOW_TASKS 1089
```

Syntax

```
void CalendarShowTasks(BOOLEAN Show)
```

Parameters

Show as BOOLEAN

Specifies the type as follows:

TRUE Show tasks
FALSE Hide tasks

Remarks

The calendar must have focus for this token to work properly.

CalendarSubscribe()

Subscribe to an internet calendar.

Token ID

```
AFTKN_CALENDAR_SUBSCRIBE or 936
```

Syntax

```
VOID CalendarSubscribe(
     WORDSTRING CalendarURL;
     WORDSTRING FolderName;
     [WORDSTRING] Description;
     [DWORD] RefreshFrequency;
     [WORDSTRING] UserName;
     [WORDSTRING] Password)
```

Parameters

CalendarURL As WIDESTRING

The URL of the internet calendar to subscribe to. For example:

```
webcal://icalx.com/public/domain/example.ics
```

FolderName As WIDESTRING

The full path of the folder to create in GroupWise to display the subscribed calendar.

Description As WIDESTRING

(Optional) A description of the internet calendar.

Password As WORDSTRING

(Optional) Password to specify by way of HTTP Basic Auth as part of the HTTP GET of the internet calendar. Note that this is only required for user name/password protected calendar.

RefreshFrequency As DWORD

(Optional) Frequency of refresh in seconds. For example:

Daily = 24 * 60 * 60
Hourly = 60 * 60
15 Minutes = 15 * 60

UserName As WORDSTRING

(Optional) User name to specify by way of HTTP Basic Auth as part of the HTTP GET of the internet calendar. Note that this is only required for user name/password protected calendars.

CalendarSubscribeDlg()

Display a dialog that enables the user to subscribe to an internet calendar.

Token ID

```
DTKN_CALENDAR_SUBSCRIBE_DLG or 146
```

Syntax

```
VOID CalendarSubscribeDlg(
     [WIDESTRING] CalendarURL;
     [WIDESTRING] FolderName;
     [WIDESTRING] Description)
```

Parameters

CalendarURL As WIDESTRING

(Optional) The URL of the internet calendar to subscribe to. For example:

```
webcal://icalx.com/public/domain/example.ics
```

FolderName As WIDESTRING

(Optional) The full path of the folder to create in GroupWise to display the subscribed calendar.

Description As WIDESTRING

(Optional) A description of the internet calendar.

CalendarSubscribePropDlg()

Displays the properties of a previously subscribed Internet calendar.

Token ID

`DTKN_CALENDAR_SUBSCRIBE_PROP_DLG or 145`

Syntax

```
VOID CalendarSubscribePropDlg()
```

CalendarSubscribeWebcal()

Subscribe to an Internet calendar.

Token ID

```
AFTKN_CALENDAR_SUBSCRIBE_WEBCAL or 937
```

Syntax

```
VOID CalendarSubscribeWebcal(
      WIDESTRING CalendarURL)
```

Parameters

CalendarURL As WIDESTRING

The URL of the internet calendar to subscribe to. For example:

```
webcal://icalx.com/public/domain/example.ics
```

Remarks

Note that this just calls CalendarSubscribeDlg() passing the specified URL.

CalendarTabPropertiesDlg()

Shows the Add Calendar Tab dialog.

Token ID

```
DTKN_CALTAB_PROPS or 129
```

Syntax

```
VOID CalendarTabPropertiesDlg()
```

Cancel()

Cancels the current view, dialog, or command.

Token ID

```
BFTKN_CANCEL or 202
```

Syntax

```
VOID Cancel()
```

CategoriesSetDlg()

Invokes the Set Categories dialog.

Token ID

```
BFTKN_CALENDAR_IS_SHOWING_NOTES 136
```

Syntax

```
VOID CategoriesSetDlg()
```

CheckboxSelect()

Selects or deselects a check box in a phone message view.

Token ID

```
AFTKN_SET_CHECKSTATE or 641
```

Syntax

```
VOID CheckboxSelect([ENUM StandardCheckBox];
                    [ENUM UserDefinedBoxName];
                       ENUM State)
```

Parameters

StandardCheckBox As ENUM

(Optional) Phone message check box, specified as follows:

0	PhoneCalled
1	PhonePleaseCall
2	PhoneWillCallAgain
3	PhoneReturnedYou
4	PhoneWantsToSeeYou
5	PhoneCameToSeeYou
6	PhoneUrgent

UserDefinedBoxName As ENUM

(Optional)

State As ENUM

If not specified, acts as a toggle:

0 Off!
1 On!

See Also

SendPhone() (page 804)

ClearAlarm()

Clears the alarm.

Token ID

```
BFTKN_CLEAR_ALARM or 978
```

Syntax

```
VOID ClearAlarm()
```

CloseDocument()

Returns the document identified by the Document Number passed in DocIDStr to the GroupWise library.

Token ID

`AFTKN_DM_CLOSE or 855`

Syntax

```
WORD CloseDocument(ANSISTRING DocIDStr)
```

Parameters

DocIDStr As ANSISTRING

String in the form Domain.PostOffice.Library:Doc#.Ver#.

Return Values

WORD

CloseWindow()

Closes the current view.

Token ID

```
BFTKN_CLOSE_WINDOW or 292
```

Syntax

```
VOID CloseWindow()
```

CollapseAll()

If the folder tree has focus, collapse either the currently selected branch or the optionally specified branch of the folder tree such that its children will no longer be displaying. If the item list has focus, Collapse all entries in a hierarchically displayed item list such that only the root entries are showing. This applies to lists viewed by discussion thread and tasklists that contain subtasks.

Token ID

```
BFTKN_COLLAPSE_ALL or 1016
```

Syntax

```
VOID CollapseAll(
     [ANSISTRING] FolderName)
```

Parameters

FolderName As ANSISTRING

The full path of the folder tree whose children should be collapsed. Note that this parameter is ignored if the token is thrown a the item list.

CollapseFolder()

Collapses the currently selected folder.

Token ID

```
BFTKN_CONTROL_MINUS 373
```

Syntax

```
VOID CollapseFolder()
```

CollapseThread()

Collapses the current thread.

Token ID

```
BFTKN_COLLAPSE_THREAD or 393
```

Syntax

```
VOID CollapseThread()
```

ColumnDlg()

Displays the dialog which allows users to specify the columns to display in the GroupWise client.

Token ID

```
DTKN_SELECT_COLUMNS or 109
```

Syntax

```
VOID ColumnDlg()
```

ContactsFolderCreate()

Create a new contact folder.

Token ID

```
DTKN_CREATE_CONTACTS_FOLDER or 147
```

Syntax

```
VOID ContactsFolderCreate()
```

ConversationPlace()

Launches the Conversation Place application.

Token ID

```
BFTKN_CONV_PLACE or 432
```

Syntax

```
VOID ConversationPlace()
```

D

This section contains information about the following tokens:

DateAbsoluteGoTo()

Changes the calendar date to a specified day, month, and year. An error message is displayed if the day, month, and year are out of range.

Token ID

```
AFTKN_GOTO_DATE_ABS or 556
```

Syntax

```
VOID DateAbsoluteGoTo([WORD Day], [WORD Month], [WORD Year])
```

Parameters

Day As WORD

(Optional) Defaults to today's date.

Month As WORD

(Optional) Defaults to today's date.

Year As WORD

(Optional) Full year string (for example, 2012). Defaults to today's date.

Remarks

NOTE: Make sure the calendar view is being displayed.

See Also

“DateParseDay()” on page 160
“DateRelativeGoTo()” on page 168

DateDifferenceDlg()

Displays the Date Difference dialog box.

Token ID

```
DTKN_DATE_DIFF or 90
```

Syntax

```
VOID DateDifferenceDlg()
```

See Also

“DateRelativeGoTo()” on page 168

DateParseDay()

Parses a date string and returns the day of the month.

Token ID

```
AFTKN_DATE_PARSE_DAY or 712
```

Syntax

```
WORD DateParseDay(ANSISTRING DateString)
```

Parameters

DateString As ANSISTRING

For example, "Jun 17 1999" returns 17.

Return Values

WORD

DateParseDayOfWeek()

Parses a date string and returns the day of the week. Note that the actual name of this token is ParseDayOfWeek() and not DateParseDayOfWeek. Use ParseDayOfWeek() instead.

Token ID

```
AFTKN_DATE_PARSE_DAY_OF_WEEK or 634
```

Syntax

```
WORD ParseDayOfWeek(ANSISTRING DateString)
```

Parameters

DateString As ANSISTRING

For example, "Jun 17 2012" returns 0 for Sunday.

Return Values

WORD

Zero-based integer (0=Sunday).

DateParseDurationHours()

Parses a date string and returns the number of hours.

Token ID

```
AFTKN_DATE_PARSE_DUR_HR or 810
```

Syntax

```
WORD DateParseDurationHours(ANSISTRING DateString)
```

Parameters

DateString As ANSISTRING

Rounds down to the closest hour. For example: "119" minutes returns 1 hour; "1 hour 30 minutes" returns 1; "2.5 days" returns 60; "60 minutes" returns 1. Limits are as follows: 255 for "minutes"; 255 for "days"; 36 for "weeks."

Return Values

WORD

DateParseDurationMinutes()

Parses a date string and returns the number of minutes.

Token ID

```
AFTKN_DATE_PARSE_DUR_MIN or 811
```

Syntax

```
WORD DateParseDurationMinutes(ANSISTRING DateString)
```

Parameters

DateString As ANSISTRING

For example, "1 hour 30 minutes" returns 30; ".25" returns 15.

Return Values

WORD

DateParseHours()

Parses a time string and returns the hour of a time string.

Token ID

```
AFTKN_DATE_PARSE_HOURS or 808
```

Syntax

```
WORD DateParseHours(ANSISTRING DateString)
```

Parameters

DateString As ANSISTRING

String in the form HH:MM[:SS][AM/PM]. For example, "10:30" returns 10; "23:59" returns 23; "11:59:55 AM" returns 11; "11:59:55 PM" returns 23.

Return Values

WORD

DateParseMinutes()

Parses a time string and returns the number of minutes of a date string.

Token ID

```
AFTKN_DATE_PARSE_MINUTES or 809
```

Syntax

```
WORD DateParseMinutes(ANSISTRING DateString)
```

Parameters

DateString As ANSISTRING

String in the form HH:MM[:SS][AM/PM]. For example, "10:30" returns 30; "23:59" returns 59.

Return Values

WORD

DateParseMonth()

Parses a date string and returns a month.

Token ID

```
AFTKN_DATE_PARSE_MONTH or 614
```

Syntax

```
WORD DateParseMonth(ANSISTRING DateString)
```

Parameters

DateString As ANSISTRING

For example, "Jun 17 2012" returns 6 for June.

Return Values

WORD

DateParseYear()

Parses a date string and returns a year.

Token ID

```
AFTKN_DATE_PARSE_YEAR or 625
```

Syntax

```
WORD DateParseYear(ANSISTRING DateString)
```

Parameters

DateString As ANSISTRING

For example, "Jun 17 2012" returns 2012.

Return Values

WORD

DateRelativeGoTo()

Sets the calendar date forward (positive) or backward (negative) relative to the current date. If no parameters are specified, the date is set to the current date.

Token ID

```
BFTKN_GOTO_DATE or 337
```

Syntax

```
VOID DateRelativeGoTo([DWORD RelativeDay];
                      [WORD RelativeMonth];
                      [WORD RelativeYear])
```

Parameters

RelativeDay As DWORD

(Optional) Number of days. Default to today's date.

RelativeMonth As WORD

(Optional) Number of months. Defaults to today's date.

RelativeYear As WORD

(Optional) Number of years. Defaults to today's date.

See Also

“DateAbsoluteGoTo()” on page 158
“DateDifferenceDlg()” on page 159

DateRelativeGoToDlg()

Displays the GoTo Date dialog box.

Token ID

```
DTKN_GOTO_DATE or 89
```

Syntax

```
VOID DateRelativeGoToDlg()
```

See Also

“DateRelativeGoTo()” on page 168

DateSetAutodateMode()

Displays the Autodate dialog box to schedule recurring appointments, tasks, or notes.

Token ID

```
DTKN_GSET_AUTODATE or 74
```

Syntax

```
VOID DateSetAutodateMode()
```

DateSetEndDateMode()

Displays the Set Date dialog box to set appointment, End Dates (if selected in Options), or task End dates.

Token ID

```
DTKN_GSET_ENDDATE or 92
```

Syntax

```
VOID DateSetEndDateMode()
```

DateSetStartDateMode()

Displays the Set Date dialog box to set the start date for appointments, tasks or notes.

Token ID

```
DTKN_GSET_STARTDATE or 75
```

Syntax

```
VOID DateSetStartDateMode()
```

DateSwitchToAutoDate()

Switches from Start Date to Autodate to schedule recurring appointments, tasks and/or notes.

Token ID

```
BFTKN_AUTODATE or 280
```

Syntax

```
VOID DateSwitchToAutoDate()
```

See Also

“DateSwitchToStartDate()” on page 176

DateSwitchToDuration()

Switches from End Date to Duration to display the length of an appointment.

Token ID

`BFTKN_DURATION or 282`

Syntax

```
VOID DateSwitchToDuration()
```

See Also

“DateSwitchToEndDate()” on page 175

DateSwitchToEndDate()

Switches from Duration to End Date to display the length of an appointment.

Token ID

```
BFTKN_ENDDATE or 281
```

Syntax

```
VOID DateSwitchToEndDate()
```

See Also

“DateSwitchToDuration()” on page 174

DateSwitchToStartDate()

Switches from Autodate to Start Date to schedule appointments, tasks, and/or notes.

Token ID

```
BFTKN_STARTDATE or 279
```

Syntax

```
VOID DateSwitchToStartDate()
```

See Also

“DateSwitchToAutoDate()” on page 173

DayColumnAdd()

Adds a new day column to the Week Calendar view (up to six days).

Token ID

```
BFTKN_COLUMN_ADD or 276
```

Syntax

```
VOID DayColumnAdd()
```

DayColumnDelete()

Deletes a day column from the Week calendar view.

Token ID

```
BFTKN_COLUMN_DELETE or 277
```

Syntax

```
VOID DayColumnDelete()
```

DBToggle()

Toggles between current and archived items.

Token ID

```
BFTKN_DB_TOGGLE or 313
```

Syntax

```
VOID DBToggle()
```

DBUseArchive()

Lists archive Mailbox, Sent Items, and Trash items.

Token ID

```
BFTKN_CHANGE_TO_ARCHIVEDB or 314
```

Syntax

```
VOID DBUseArchive()
```

DBUsePrimary()

Lists current Mailbox, Sent Items, and Trash items.

Token ID

```
BFTKN_CHANGE_TO_USERDB or 315
```

Syntax

```
VOID DBUsePrimary()
```

Delete()

Deletes selected text, selected items, or a character at the insertion point.

Token ID

```
BFTKN_DELETE or 216
```

Syntax

```
VOID Delete()
```

DeleteAndEmpty()

Deletes and purges the currently selected item(s).

Token ID

```
BFTKN_DELETE_AND_EMPTY or 1034
```

Syntax

```
VOID DeleteAndEmpty()
```

Remarks

The user is prompted with a confirmation dialog before the actual delete and purge take place.

DeleteCharPrevious()

Deletes selected text, or one character to the left of the insertion point.

Token ID

```
BFTKN_BACKSPACE or 211
```

Syntax

```
VOID DeleteCharPrevious()
```

DeleteDocument()

Deletes the document identified by the Document Number in DocIDStr from the GroupWise library.

Token ID

```
AFTKN_DM_DELETE or 863
```

Syntax

```
VOID DeleteDocument(ANSISTRING DocIDStr)
```

Parameters

DocIDStr As ANSISTRING

The document number of the document to delete.

Remarks

IMPORTANT: DocIDStr in the form Domain.PostOffice.Library:Doc#.Ver# for ObjectAPI reference: LibraryID: Doc#.Ver#. The DocIDStr is a subset of the object DocumentVersion.ODMADocumentID in the ObjectAPI.

DeleteWordLeft()

Deletes the word at the insertion point, to the left of the insertion point (if it is after the last character), or to the right of the insertion point (if it is before the first character).

Token ID

```
BFTKN_DELETEWORDLEFT or 218
```

Syntax

```
VOID DeleteWordLeft()
```

DeleteWordRight()

Deletes all text right of the insertion line to the end of the line.

Token ID

```
BFTKN_DELETEWORDRIGHT or 219
```

Syntax

```
VOID DeleteWordRight()
```

DialPeople()

Calls the list of phone numbers using the default dialer specified by the user.

Token IDs

```
BFTKN_DIAL_PEOPLE or 1340
```

Syntax

```
DialPeople(ANSISTRING Optional Comment,
           BOOLEAN IsDefault,
           ANSISTRING TelephoneNumber,
           ANSISTRING UserDiaplayName,
           ANSISTRING UserEMailAddress,
           ENUM WHichTelephone);
```

Parameters

Optional comment

(Optional) String containing anything

IsDefault

Enumeration telling if this is the default phone number.

```
"No!0|"
"Yes!1}"
```

TelephoneNumber

String containing the telephone number.

UserDisplayName

(Optional) String containing this user’s display name

UserEMailAddress

(Optional) String containing this user’s email address.

WhichTelephone

(Optional) Enumeration describing which telephone number this is.

```
"AssisntantPhone!870|"
"CallbackPhone!863|"
"CarPhone!866|"
"CellPhoneNum!802|"
"FaxPhoneNum!810|"
"Home2Phone!871|"
"HomePhoneNum!817|"
"IsdnPhone!869|"
"MainCompanyPhone!875|"
"Office2Phone!864|"
"OfficePhoneNum!829|"
"OtherPhone!867|"
"PagerPhoneNum!833|"
"Phone!8|"
"PhoneNum!835|"
"RadioPhone!865|"
"SkypePhone!904|"
"TelexPhone!868};"
```

Remarks

The group containing `IsDefault` through `WhichTelephone` may be repeated up to 10 times

DialSender()

Causes Conversation Place to dial the phone number of the sender of the active item.

Token ID

```
BFTKN_DIAL_SENDER or 433
```

Syntax

```
VOID DialSender()
```

DisplayProfile()

Displays the document profile for the document in the GroupWise library identified by the Document Number in DocIDStr.

Token ID

`AFTKN_DM_DISPLAY_PROFILE or 860`

Syntax

```
VOID DisplayProfile(ANSISTRING DocIDStr)
```

Parameters

DocIDStr As ANSISTRING

Document number of the document to display the profile for.

Remarks

IMPORTANT: DocIDStr in the form Domain.PostOffice.Library:Doc#.Ver# for ObjectAPI reference: LibraryID: Doc#.Ver#. The DocIDStr is a subset of the object DocumentVersion.ODMADocumentID in the ObjectAPI.

DisplaySettingsSave()

Save the currently active display settings into the user settings.

Token ID

```
BFTKN_DISPLAY_SETTINGS_SAVE or 1241
```

Syntax

```
VOID DisplaySettingsSave()
```

Remarks

Settings are normally saved when a user switches off of a folder or when the client shuts down. This token allows the settings to be saved without waiting for either of these events to occur.

DisplaySettingsSelect()

Displays the Select Display Settings dialog box.

Token ID

```
DTKN_DISPLAY_SETTINGS_SELECT or 126
```

Syntax

```
VOID DisplaySettingsSelect()
```

DisplaySettingsSend()

Export the currently active settings to a file and attach it to an email to be sent to another user.

Token ID

```
BFTKN_DISPLAY_SETTINGS_SEND or 1277
```

Syntax

```
VOID DisplaySettingsSend()
```

Remarks

Settings are normally saved when a user switches off of a folder or when the client shuts down. This token allows the settings to be saved without waiting for either of these events to occur.

DisplaySettingsSet()

Customizes the display appearance of folder contents in the GroupWise client. Also lets you create named display settings. See also DisplaySettingsSetEx(), DisplaySettingsSetEx2(), DisplaySettingsSetEx3(), and DisplaySettingsSetEx4().

Token ID

```
AFTKN_DISPLAY_SETTINGS_SET or 843
```

Syntax

```
VOID DisplaySettingsSet(ANSISTRING FolderName;
                        [ANSISTRING SettingsName];
                        [ANSISTRING SettingsDesc];
                        [ENUM ViewMode];
                        [ENUM ItemSourceIncoming];
                        [ENUM ItemSourceOutgoing];
                        [ENUM ItemSourcePersonal];
                        [ENUM ItemSourceDraft];
                        [ENUM SortKey];
                        [ANSISTRING SortFieldName];
                        [ENUM SortOrder];
                        [ANSISTRING UserID];
                        [ENUM Column00];
                        [ANSISTRING ColumnFieldName00];
                        [WORD ColumnWidth00];
                        [ENUM Column01];
                        [ANSISTRING ColumnFieldName01];
                        [WORD ColumnWidth01];
                        [ENUM Column02];
                        [ANSISTRING ColumnFieldName02];
                        [WORD ColumnWidth02];
                        [ENUM Column03];
                        [ANSISTRING ColumnFieldName03];
                        [WORD ColumnWidth03];
                        [ENUM Column04];
                        [ANSISTRING ColumnFieldName04];
                        [WORD ColumnWidth04];
                        [ENUM Column05];
                        [ANSISTRING ColumnFieldName05];
                        [WORD ColumnWidth05];
                        [ENUM Column06];
                        [ANSISTRING ColumnFieldName06];
                        [WORD ColumnWidth06];
                        [ENUM Column07];
                        [ANSISTRING ColumnFieldName07];
                        [WORD ColumnWidth07];
                        [ENUM Column08];
                        [ANSISTRING ColumnFieldName08];
                        [WORD ColumnWidth08];
                        [ENUM Column09];
                        [ANSISTRING ColumnFieldName09];
                        [WORD ColumnWidth09];
                        [ENUM Column10];
                        [ANSISTRING ColumnFieldName10];
                        [WORD ColumnWidth10];
```

```
[ENUM Column11];
[ANSISTRING ColumnFieldName11];
[WORD ColumnWidth11];
[ENUM Column12]
[ANSISTRING ColumnFieldName12];
[WORD ColumnWidth12];
[ENUM Column13];
[ANSISTRING ColumnFieldName13];
[WORD ColumnWidth13];
[ENUM Column14];
[ANSISTRING ColumnFieldName14];
[WORD ColumnWidth14];
[ENUM Column15];
[ANSISTRING ColumnFieldName15];
[WORD ColumnWidth15];
[ENUM Column16];
[ANSISTRING ColumnFieldName16];
[WORD ColumnWidth16];
[ENUM Column17];
[ANSISTRING ColumnFieldName17];
[WORD ColumnWidth17];
[ENUM Column18];
[ANSISTRING ColumnFieldName18];
[WORD ColumnWidth18];
[ENUM Column19];
[ANSISTRING ColumnFieldName19];
    [WORD ColumnWidth19])
```

Parameters

FolderName As ANSISTRING

The name of the folder.

SettingsName As ANSISTRING

(Optional) The name for the set of display settings.

SettingsDesc As ANSISTRING

(Optional) The description for the set of display settings.

ViewMode As ENUM

(Optional) Specifies the view mode as follows:

1	Calendar
2	Conversation
3	Details

ItemSourceIncoming As ENUM

(Optional)

0 No
1 Yes

ItemSourceOutgoing As ENUM

(Optional)

0 No
1 Yes

ItemSourcePersonal As ENUM

(Optional)

0 No
1 Yes

ItemSourceDraft As ENUM

(Optional)

0 No
1 Yes

SortKey As ENUM

(Optional) Specifies the sort key as follows:

600	AcceptedCnt
112	AssignedDate
625	Author
4	Authority
3	BC
126	BeginDateTime
2	CC
6	Caller
7	Company
603	CompletedCnt
602	CopyType
139	CreateDate
626	Creator
627	CurrentVersion
176	Date
606	DeletedCnt
605	DeletedDate
507	DeliveredDate
656	DocCreatedDate
0	DocType
608	DocumentID
629	DocumentType
148	DueDate

151	EndDateTime
628	Filename
1	From
169	ItemType
613	Label
614	Library
647	LibraryId
10	Message
633	Modified
331	NamedField (for custom fields)
703	NamedFldByte
702	NamedFldDate
706	NamedFldSDWord
704	NamedFldSWord
700	NamedFldText
707	NamedFldUDWord
705	NamedFldUWord
701	NamedFldWrdStr
630	OfficialVersion
615	OpenCnt
8	Phone
5	Place
616	RecipientsCnt
617	RepliedCnt
631	RetrievedBy
632	RetrievedDate
657	SharedName
618	Size
619	Source
646	Subclass
9	Subject
646	SubType
227	TaskCategory
230	TaskPriority
351	ThreadStatus

0	To
621	TransferCnt
622	UndeliveredCnt
623	Version
634	VersionCreator
635	VersionDate
636	VersionDescription
267	ViewName

SortFieldName As ANSISTRING

(Optional)

SortOrder As ENUM

(Optional)

111 Ascending
146 Descending

UserID As ANSISTRING

(Optional) User ID of the mailbox where the display settings are being set.

Column00 As ENUM

(Optional) Use the same enumerated values as Sortkey as ENUM.

ColumnFieldName00 As ANSISTRING

(Optional) Name of the column.

ColumnWidth00 As WORD

(Optional) Width of the column

Column01 As ENUM

(Optional) Use the same enumerated values as Sortkey as ENUM.

ColumnFieldName01 As ANSISTRING

(Optional) Name of the column.

ColumnWidth01 As WORD

(Optional) Width of the column.

Column02 As ENUM

(Optional) Use the same enumerated values as Sortkey as ENUM.

ColumnFieldName02 As ANSISTRING

(Optional) Name of the column.

ColumnWidth02 As WORD

(Optional) Width of the column.

Column03 As ENUM

(Optional) Use the same enumerated values as Sortkey as ENUM.

ColumnFieldName03 As ANSISTRING

(Optional) Name of the column.

ColumnWidth03 As WORD

(Optional) Width of the column.

Column04 As ENUM

(Optional) Use the same enumerated values as Sortkey as ENUM.

ColumnFieldName04 As ANSISTRING

(Optional) Name of the column.

ColumnWidth04 As WORD

(Optional) Width of the column.

Column05 As ENUM

(Optional) Use the same enumerated values as Sortkey as ENUM.

ColumnFieldName05 As ANSISTRING

(Optional) Name of the column.

ColumnWidth05 As WORD

(Optional) Width of the column.

Column06 As ENUM

(Optional) Use the same enumerated values as Sortkey as ENUM.

ColumnFieldName06 As ANSISTRING

(Optional) Name of the column.

ColumnWidth06 As WORD

(Optional) Width of the column.

Column07 As ENUM

(Optional) Use the same enumerated values as Sortkey as ENUM.

ColumnFieldName07 As ANSISTRING

(Optional) Name of the column.

ColumnWidth07 As WORD

(Optional) Width of the column.

Column08 As ENUM

(Optional) Use the same enumerated values as Sortkey as ENUM.

ColumnFieldName08 As ANSISTRING

(Optional) Name of the column.

ColumnWidth08 As WORD

(Optional) Width of the column.

Column09 As ENUM

(Optional) Use the same enumerated values as Sortkey as ENUM.

ColumnFieldName09 As ANSISTRING

(Optional) Name of the column.

ColumnWidth09 As WORD

(Optional) Width of the column.

Column10 As ENUM

(Optional) Use the same enumerated values as Sortkey as ENUM.

ColumnFieldName10 As ANSISTRING

(Optional) Name of the column.

ColumnWidth10 As WORD

(Optional) Width of the column.

Column11 As ENUM

(Optional) Use the same enumerated values as Sortkey as ENUM.

ColumnFieldName11 As ANSISTRING

(Optional) Name of the column.

ColumnWidth11 As WORD

(Optional) Width of the column.

Column12 As ENUM

(Optional) Use the same enumerated values as Sortkey as ENUM.

ColumnFieldName12 As ANSISTRING

(Optional) Name of the column.

ColumnWidth12 As WORD

(Optional) Width of the column.

Column13 As ENUM

(Optional) Use the same enumerated values as Sortkey as ENUM.

ColumnFieldName13 As ANSISTRING

(Optional) Name of the column.

ColumnWidth13 As WORD

(Optional) Width of the column.

Column14 As ENUM

(Optional) Use the same enumerated values as Sortkey as ENUM.

ColumnFieldName14 As ANSISTRING

(Optional) Name of the column.

ColumnWidth14 As WORD

(Optional) Width of the column.

Column15 As ENUM

(Optional) Use the same enumerated values as Sortkey as ENUM.

ColumnFieldName15 As ANSISTRING

(Optional) Name of the column.

ColumnWidth15 As WORD

(Optional) Width of the column.

Column16 As ENUM

(Optional) Use the same enumerated values as Sortkey as ENUM.

ColumnFieldName16 As ANSISTRING

(Optional) Name of the column.

ColumnWidth16 As WORD

(Optional) Width of the column.

Column17 As ENUM

(Optional) Use the same enumerated values as Sortkey as ENUM.

ColumnFieldName17 As ANSISTRING

(Optional) Name of the column.

ColumnWidth17 As WORD

(Optional) Width of the column.

Column18 As ENUM

(Optional) Use the same enumerated values as Sortkey as ENUM.

ColumnFieldName18 As ANSISTRING

(Optional) Name of the column.

ColumnWidth18 As WORD

(Optional) Width of the column.

Column19 As ENUM

(Optional) Use the same enumerated values as Sortkey as ENUM.

ColumnFieldName19 As ANSISTRING

(Optional) Name of the column.

ColumnWidth19 As WORD

(Optional) Width of the column.

DisplaySettingsSetEx()

Customizes the display appearance of folder contents in the GroupWise client. It is also possible to create named display settings. See also DisplaySettingsSet(), DisplaySettingsSetEx2(), DisplaySettingsSetEx3(), and DisplaySettingsSetEx4().

Token ID

AFTKN_DISPLAY_SETTINGS_SET_EX or 887

Syntax

```
VOID DisplaySettingsSetEx(ANSISTRING FolderName;
                          [ANSISTRING SettingsName];
                          [ANSISTRING SettingsDesc];
                          [ENUM ViewMode];
                          [ENUM ItemSourceIncoming];
                          [ENUM ItemSourceOutgoing];
                          [ENUM ItemSourcePersonal];
                          [ENUM ItemSourceDraft];
                          [ENUM ItemTypeAppointment];
                          [ENUM ItemTypeDocument];
                          [ENUM ItemTypeMail];
                          [ENUM ItemTypeNote];
                          [ENUM ItemTypePhone];
                          [ENUM ItemTypeTask];
                          [ENUM SortKey];
                          [ANSISTRING SortFieldName];
                          [ENUM SortOrder];
                          [ANSISTRING UserID];
                          [ENUM Column00];
                          [ANSISTRING ColumnFieldName00];
                          [WORD ColumnWidth00];
                          [ENUM Column01];
                          [ANSISTRING ColumnFieldName01];
                          [WORD ColumnWidth01];
                          [ENUM Column02];
                          [ANSISTRING ColumnFieldName02];
                          [WORD ColumnWidth02];
                          [ENUM Column03];
                          [ANSISTRING ColumnFieldName03];
                          [WORD ColumnWidth03];
                           [ENUM Column04];
                           [ANSISTRING ColumnFieldName04];
                           [WORD ColumnWidth04];
                           [ENUM Column05];
                           [ANSISTRING ColumnFieldName05];
                           [WORD ColumnWidth05];
                           [ENUM Column06];
                           [ANSISTRING ColumnFieldName06];
                           [WORD ColumnWidth06];
                           [ENUM Column07];
                           [ANSISTRING ColumnFieldName07];
                           [WORD ColumnWidth07];
                           [ENUM Column08];
                           [ANSISTRING ColumnFieldName08];
                           [WORD ColumnWidth08];
```

```
[ENUM Column09];
[ANSISTRING ColumnFieldName09];
[WORD ColumnWidth09];
[ENUM Column10];
[ANSISTRING ColumnFieldName10];
[WORD ColumnWidth10];
[ENUM Column11];
[ANSISTRING ColumnFieldName11];
[WORD ColumnWidth11];
[ENUM Column12];
[ANSISTRING ColumnFieldName12];
[WORD ColumnWidth12];
[ENUM Column13];
[ANSISTRING ColumnFieldName13];
[WORD ColumnWidth13];
[ENUM Column14];
[ANSISTRING ColumnFieldName14];
[WORD ColumnWidth14];
[ENUM Column15];
[ANSISTRING ColumnFieldName15];
[WORD ColumnWidth15];
[ENUM Column16];
[ANSISTRING ColumnFieldName16];
[WORD ColumnWidth16];
[ENUM Column17];
[ANSISTRING ColumnFieldName17];
[WORD ColumnWidth17];
[ENUM Column18];
[ANSISTRING ColumnFieldName18];
[WORD ColumnWidth18];
[ENUM Column19];
[ANSISTRING ColumnFieldName19];
    [WORD ColumnWidth19])
```

Parameters

FolderName As ANSISTRING

The name of the folder.

SettingsName As ANSISTRING

(Optional) Name for the set of display settings.

SettingsDesc As ANSISTRING

(Optional) Description for the set of display settings.

ViewMode As ENUM

(Optional) Specifies the view mode as follows:

1	Calendar
2	Conversation
3	Details

ItemSourceIncoming As ENUM

(Optional)

0 No
1 Yes

ItemSourceOutgoing As ENUM

(Optional)

0 No
1 Yes

ItemSourcePersonal As ENUM

(Optional)

0 No
1 Yes

ItemSourceDraft As ENUM

(Optional)

0 No
1 Yes

ItemTypeAppointment As ENUM

(Optional)

0 No
1 Yes

ItemTypeDocument As ENUM

(Optional)

0 No
1 Yes

ItemTypeMail As ENUM

(Optional)

0 No
1 Yes

ItemTypeNote As ENUM

(Optional)

0 No
1 Yes

ItemTypePhone As ENUM

(Optional)

0 No
1 Yes

ItemTypeTask As ENUM

(Optional)

0 No
1 Yes

SortKey As ENUM

(Optional)

600 AcceptedCnt
112 AssignedDate
625 Author
4 Authority
126 BeginDateTime
6 Caller
7 Company
603 CompletedCnt
602 CopyType
139 CreateDate
626 Creator
176 Date
606 DeletedCnt
605 DeletedDate
507 DeliveredDate
656 DocCreatedDate
608 DocumentID
629 DocumentType
148 DueDate
151 EndDateTime
1 From
169 ItemType
614 Library
647 LibraryId
10 Message
633 Modified
331 NamedField (for custom fields)
703 NamedFldByte
702 NamedFldDate
706 NamedFldSDWord
704 NamedFldSWord
700 NamedFldText
707 NamedFldUDWord
705 NamedFldUWord
701 NamedFldWrdStr
615 OpenedCnt
8 Phone
5 Place
616 RecipientsCnt
617 RepliedCnt
657 SharedName
618 Size
619 Source
646 Subclass
9 Subject

227 TaskCategory
230 TaskPriority
351 ThreadStatus
0 To
622 UndeliveredCnt
623 Version
634 VersionCreator
635 VersionDate
636 VersionDescription
267 ViewName

SortFieldName As ANSISTRING

(Optional) Name of the field to sort on.

SortOrder As ENUM

(Optional)

111 Ascending
146 Descending

UserID As ANSISTRING

(Optional) User ID of the mailbox where the display settings are being set.

Column00 As ENUM

(Optional) See the values for SortKey.

ColumnFieldName00 As ANSISTRING

(Optional) Column name.

ColumnWidth00 As WORD

(Optional) Column width.

Column01 As ENUM

(Optional) See the values for SortKey.

ColumnFieldName01 As ANSISTRING

(Optional) Column name.

ColumnWidth01 As WORD

(Optional) Column width.

Column02 As ENUM

(Optional) See the values for SortKey.

ColumnFieldName02 As ANSISTRING

(Optional) Column name.

ColumnWidth02 As WORD

(Optional) Column width.

Column03 As ENUM

(Optional) See the values for SortKey.

ColumnFieldName03 As ANSISTRING

(Optional) Column name.

ColumnWidth03 As WORD

(Optional) Column width.

Column04 As ENUM

(Optional) See the values for SortKey.

ColumnFieldName04 As ANSISTRING

(Optional) Column name.

ColumnWidth04 As WORD

(Optional) Column width.

Column05 As ENUM

(Optional) See the values for SortKey.

ColumnFieldName05 As ANSISTRING

(Optional) Column name.

ColumnWidth05 As WORD

(Optional) Column width.

Column06 As ENUM

(Optional) See the values for SortKey.

ColumnFieldName06 As ANSISTRING

(Optional) Column name.

ColumnWidth06 As WORD

(Optional) Column width.

Column07 As ENUM

(Optional) See the values for SortKey.

ColumnFieldName07 As ANSISTRING

(Optional) Column name.

ColumnWidth07 As WORD

(Optional) Column width.

Column08 As ENUM

(Optional) See the values for SortKey.

ColumnFieldName08 As ANSISTRING

(Optional) Column name.

ColumnWidth08 As WORD

(Optional) Column width.

Column09 As ENUM

(Optional) See the values for SortKey.

ColumnFieldName09 As ANSISTRING

(Optional) Column name.

ColumnWidth09 As WORD

(Optional) Column width.

Column10 As ENUM

(Optional) See the values for SortKey.

ColumnFieldName10 As ANSISTRING

(Optional) Column name.

ColumnWidth10 As WORD

(Optional) Column width.

Column11 As ENUM

(Optional) See the values for SortKey.

ColumnFieldName11 As ANSISTRING

(Optional) Column name.

ColumnWidth11 As WORD

(Optional) Column width.

Column12 As ENUM

(Optional) See the values for SortKey.

ColumnFieldName12 As ANSISTRING

(Optional) Column name.

ColumnWidth12 As WORD

(Optional) Column width.

Column13 As ENUM

(Optional) See the values for SortKey.

ColumnFieldName13 As ANSISTRING

(Optional) Column name.

ColumnWidth13 As WORD

(Optional) Column width.

Column14 As ENUM

(Optional) See the values for SortKey.

ColumnFieldName14 As ANSISTRING

(Optional) Column name.

ColumnWidth14 As WORD

(Optional) Column width.

Column15 As ENUM

(Optional) See the values for SortKey.

ColumnFieldName15 As ANSISTRING

(Optional) Column name.

ColumnWidth15 As WORD

(Optional) Column width.

Column16 As ENUM

(Optional) See the values for SortKey.

ColumnFieldName16 As ANSISTRING

(Optional) Column name.

ColumnWidth16 As WORD

(Optional) Column width.

Column17 As ENUM

(Optional) See the values for SortKey.

ColumnFieldName17 As ANSISTRING

(Optional) Column name.

ColumnWidth17 As WORD

(Optional) Column width.

Column18 As ENUM

(Optional) See the values for SortKey.

ColumnFieldName18 As ANSISTRING

(Optional) Column name.

ColumnWidth18 As WORD

(Optional) Column width.

Column19 As ENUM

(Optional) See the values for SortKey.

ColumnFieldName19

(Optional) Column name.

ColumnWidth19

(Optional) Column width.

DisplaySettingsSetEx2()

Customizes the display appearance of folder contents in the GroupWise client. It is also possible to create named display settings. See also DisplaySettingsSetEx(), DisplaySettingsSetEx(), DisplaySettingsSetEx3(), and DisplaySettingsSetEx4().

Token ID

```
AFTKN_DISPLAY_SETTINGS_SET_EX2 or 920
```

Syntax

```
VOID DisplaySettingsSetEx2(ANSISTRING FolderName;
                           [ANSISTRING SettingsName];
                           [ANSISTRING SettingsDesc];
                           [ENUM QuickViewerPerFolder];
                           [ENUM ShowQuickViewer];
                           [ENUM ViewMode];
                           [ENUM ItemSourceIncoming];
                           [ENUM ItemSourceOutgoing];
                           [ENUM ItemSourcePersonal];
                           [ENUM ItemSourceDraft];
                           [ENUM ItemTypeAppointment];
                           [ENUM ItemTypeDocument];
                           [ENUM ItemTypeMail];
                           [ENUM ItemTypeNote];
                           [ENUM ItemTypePhone];
                           [ENUM ItemTypeTask];
                           [ENUM ContactTypePerson];
                           [ENUM ContactTypeGroup];
                           [ENUM ContactTypeResource];
                           [ENUM ContactTypeOrganization];
                           [ENUM HideNonChecklistItems];
                           [ENUM SortKey];
                           [ANSISTRING SortFieldName];
                           [ENUM SortOrder];
                           [ANSISTRING UserID];
                           [ENUM Column00];
                           [ANSISTRING ColumnFieldName00];
                           [WORD ColumnWidth00];
                           [ENUM Column01];
                           [ANSISTRING ColumnFieldName01];
                           [WORD ColumnWidth01];
                           [ENUM Column02];
                           [ANSISTRING ColumnFieldName02];
                           [WORD ColumnWidth02];
                           [ENUM Column03];
                           [ANSISTRING ColumnFieldName03];
                           [WORD ColumnWidth03];
                           [ENUM Column04];
                           [ANSISTRING ColumnFieldName04];
                           [WORD ColumnWidth04];
                           [ENUM Column05];
                           [ANSISTRING ColumnFieldName05];
                           [WORD ColumnWidth05];
                           [ENUM Column06];
                           [ANSISTRING ColumnFieldName06];
```

```
[WORD ColumnWidth06];
[ENUM Column07];
[ANSISTRING ColumnFieldName07];
[WORD ColumnWidth07];
[ENUM Column08];
[ANSISTRING ColumnFieldName08];
[WORD ColumnWidth08];
[ENUM Column09];
[ANSISTRING ColumnFieldName09];
[WORD ColumnWidth09];
[ENUM Column10];
[ANSISTRING ColumnFieldName10];
[WORD ColumnWidth10];
[ENUM Column11];
[ANSISTRING ColumnFieldName11];
[WORD ColumnWidth11];
[ENUM Column12];
[ANSISTRING ColumnFieldName12];
[WORD ColumnWidth12];
[ENUM Column13];
[ANSISTRING ColumnFieldName13];
[WORD ColumnWidth13];
[ENUM Column14];
[ANSISTRING ColumnFieldName14];
[WORD ColumnWidth14];
[ENUM Column15];
[ANSISTRING ColumnFieldName15];
[WORD ColumnWidth15];
[ENUM Column16];
[ANSISTRING ColumnFieldName16];
[WORD ColumnWidth16];
[ENUM Column17];
[ANSISTRING ColumnFieldName17];
[WORD ColumnWidth17];
[ENUM Column18];
[ANSISTRING ColumnFieldName18];
[WORD ColumnWidth18];
[ENUM Column19];
[ANSISTRING ColumnFieldName19];
    [WORD ColumnWidth19])
```

Parameters

FolderName As ANSISTRING

The name of the folder.

SettingsName As ANSISTRING

(Optional) Name of the set of display settings.

SettingsDesc As ANSISTRING

(Optional) Description of the set of display settings.

QuickViewerPerFolder As ENUM

(Optional) Specifies whether to quick view the folder:

0 No
1 Yes

ShowQuickViewer As ENUM

(Optional) Specifies whether to show the quick viewer:

0 No
1 Yes

ViewMode As ENUM

(Optional) Specifies the view mode as follows:

1	Calendar
2	Conversation
3	Details
4	Checklist

ItemSourceIncoming As ENUM

(Optional)

0 No
1 Yes

ItemSourceOutgoing As ENUM

(Optional)

0 No
1 Yes

ItemSourcePersonal As ENUM

(Optional)

0 No
1 Yes

ItemSourceDraft As ENUM

(Optional)

0 No
1 Yes

ItemTypeAppointment As ENUM

(Optional)

0 No
1 Yes

ItemTypeDocument As ENUM

(Optional)

0 No
1 Yes

ItemTypeMail As ENUM

(Optional)

0 No
1 Yes

ItemTypeNote As ENUM

(Optional)

0 No
1 Yes

ItemTypePhone As ENUM

(Optional)

0 No
1 Yes

ItemTypeTask As ENUM

(Optional)

0 No
1 Yes

ContactTypePerson As ENUM

(Optional)

0 No
1 Yes

ContactTypeGroup As ENUM

(Optional)

0 No
1 Yes

ContactTypeResource As ENUM

(Optional)

0 No
1 Yes

ContactTypeOrganization As ENUM

(Optional)

0 No
1 Yes

HideNonChecklistItems As ENUM

(Optional)

0 No
1 Yes

SortKey As ENUM

(Optional)

600 AcceptedCnt
800 AdditionalRoute

112 AssignedDate
625 Author
4 Authority
126 BeginDateTime
847 Birthday
6 Caller
361 Category
802 CellPhoneNum
803 City
804 Comment
7 Company
603 CompletedCnt
825 ContactName
602 CopyType
805 Country
139 CreateDate
626 Creator
176 Date
606 DeletedCnt
605 DeletedDate
507 DeliveredDate
806 Department
656 DocCreatedDate
608 DocumentID
629 DocumentType
807 Domain
148 DueDate
808 EmailAddress
809 EmailType
151 EndDateTime
810 FaxPhoneNum
811 FirstName
1 From
812 Greeting
813 GUID
814 HomeAddress
815 HomeCity
816 HomeCountry
817 HomePhoneNum
818 HomeState
819 HomeZipCode
820 IDomain
801 IMScreenName
169 ItemType
821 LastName
822 LastReferenceDate
614 Library

647 LibraryId
823 Mailstop
824 MiddleName
633 Modified
184 Name
703 NamedFldByte
702 NamedFldDate
706 NamedFldSDWord
704 NamedFldSWord
700 NamedFldText
707 NamedFldUDWord
705 NamedFldUWord
701 NamedFldWrdStr
826 NDSDistinguishedName
827 NetworkID
828 OfficeAddress
829 OfficePhoneNum
830 OfficeWebsite
615 OpenedCnt
831 Organization
832 Owner
833 PagerPhoneNum
834 PersonalWebsite
8 Phone
835 PhoneNum
5 Place
836 PostOffice
837 Prefix
616 RecipientsCnt
838 ReferenceCount
617 RepliedCnt
657 SharerName
618 Size
619 Source
839 State
646 Subclass
9 Subject
227 TaskCategory
230 TaskPriority
351 ThreadStatus
841 Title
0 To
842 UserID
622 UndeliveredCnt
623 Version
634 VersionCreator
635 VersionDate

636 VersionDescription
267 ViewName
843 ZipCode

SortFieldName As ANSISTRING

(Optional) Name of the field to sort on.

SortOrder As ENUM

(Optional)

111 Ascending
146 Descending

UserID As ANSISTRING

(Optional) User ID of the mailbox where the display settings are being set.

Column00 As ENUM

(Optional) See the values for SortKey.

ColumnFieldName00 As ANSISTRING

(Optional) Column name.

ColumnWidth00 As WORD

(Optional) Column width.

Column01 As ENUM

(Optional) See the values for SortKey.

ColumnFieldName01 As ANSISTRING

(Optional) Column name.

ColumnWidth01 As WORD

(Optional) Column width.

Column02 As ENUM

(Optional) See the values for SortKey.

ColumnFieldName02 As ANSISTRING

(Optional) Column name.

ColumnWidth02 As WORD

(Optional) Column width.

Column03 As ENUM

(Optional) See the values for SortKey.

ColumnFieldName03 As ANSISTRING

(Optional) Column name.

ColumnWidth03 As WORD

(Optional) Column width.

Column04 As ENUM

(Optional) See the values for SortKey.

ColumnFieldName04 As ANSISTRING

(Optional) Column name.

ColumnWidth04 As WORD

(Optional) Column width.

Column05 As ENUM

(Optional) See the values for SortKey.

ColumnFieldName05 As ANSISTRING

(Optional) Column name.

ColumnWidth05 As WORD

(Optional) Column width.

Column06 As ENUM

(Optional) See the values for SortKey.

ColumnFieldName06 As ANSISTRING

(Optional) Column name.

ColumnWidth06 As WORD

(Optional) Column width.

Column07 As ENUM

(Optional) See the values for SortKey.

ColumnFieldName07 As ANSISTRING

(Optional) Column name.

ColumnWidth07 As WORD

(Optional) Column width.

Column08 As ENUM

(Optional) See the values for SortKey.

ColumnFieldName08 As ANSISTRING

(Optional) Column name.

ColumnWidth08 As WORD

(Optional) Column width.

Column09 As ENUM

(Optional) See the values for SortKey.

ColumnFieldName09 As ANSISTRING

(Optional) Column name.

ColumnWidth09 As WORD

(Optional) Column width.

Column10 As ENUM

(Optional) See the values for SortKey.

ColumnFieldName10 As ANSISTRING

(Optional) Column name.

ColumnWidth10 As WORD

(Optional) Column width.

Column11 As ENUM

(Optional) See the values for SortKey.

ColumnFieldName11 As ANSISTRING

(Optional) Column name.

ColumnWidth11 As WORD

(Optional) Column width.

Column12 As ENUM

(Optional) See the values for SortKey.

ColumnFieldName12 As ANSISTRING

(Optional) Column name.

ColumnWidth12 As WORD

(Optional) Column width.

Column13 As ENUM

(Optional) See the values for SortKey.

ColumnFieldName13 As ANSISTRING

(Optional) Column name.

ColumnWidth13 As WORD

(Optional) Column width.

Column14 As ENUM

(Optional) See the values for SortKey.

ColumnFieldName14 As ANSISTRING

(Optional) Column name.

ColumnWidth14 As WORD

(Optional) Column width.

Column15 As ENUM

(Optional) See the values for SortKey.

ColumnFieldName15 As ANSISTRING

(Optional)

ColumnWidth15 As WORD

(Optional)

Column16 As ENUM

(Optional) See the values for SortKey.

ColumnFieldName16 As ANSISTRING

(Optional) Column name.

ColumnWidth16 As WORD

(Optional) Column width.

Column17 As ENUM

(Optional) See the values for SortKey.

ColumnFieldName17 As ANSISTRING

(Optional) Column name.

ColumnWidth17 As WORD

(Optional) Column width.

Column18 As ENUM

(Optional) See the values for SortKey.

ColumnFieldName18 As ANSISTRING

(Optional) Column name.

ColumnWidth18 As WORD

(Optional) Column width.

Column19 As ENUM

(Optional) See the values for SortKey.

ColumnFieldName19

(Optional) Column name.

ColumnWidth19

(Optional) Column width.

DisplaySettingsSetEx3()

Customizes the display appearance of folder contents in the GroupWise client. See also DisplaySettingsSet(), DisplaySettingsSetEx(), DisplaySettingsSetEx2(), and DisplaySettingsSetEx4().

Token ID

```
AFTKN_DISPLAY_SETTINGS_SET_EX3 or 923
```

Syntax

```
VOID DisplaySettingsSetEx3(ANSISTRING FolderName;
                           [ANSISTRING SettingsName];
                           [ANSISTRING SettingsDesc];
                           [ENUM QuickViewerPerFolder];
                           [ENUM ShowQuickViewer];
                           [ENUM ViewMode];
                           [ENUM ItemSourceIncoming];
                           [ENUM ItemSourceOutgoing];
                           [ENUM ItemSourcePersonal];
                           [ENUM ItemSourceDraft];
                           [ENUM ItemTypeAppointment];
                           [ENUM ItemTypeDocument];
                           [ENUM ItemTypeMail];
                           [ENUM ItemTypeNote];
                           [ENUM ItemTypePhone];
                           [ENUM ItemTypeTask];
                           [ENUM ContactTypePerson];
                           [ENUM ContactTypeGroup];
                           [ENUM ContactTypeResource];
                           [ENUM ContactTypeOrganization];
                           [DWORD FilterDaysForward];
                           [DWORD FilterDaysBackward];
                           [ENUM ShowGroupLabels];
                           [ENUM NoColumnSummary];
                           [ENUM HideNonChecklistItems];
                           [ENUM SortKey];
                           [ANSISTRING SortFieldName];
                           [ENUM SortOrder];
                           [ANSISTRING UserID];
                           [ENUM Column00];
                           [ANSISTRING ColumnFieldName00];
                           [WORD ColumnWidth00];
                           [ENUM Column01];
                           [ANSISTRING ColumnFieldName01];
                           [WORD ColumnWidth01];
                           [ENUM Column02];
                           [ANSISTRING ColumnFieldName02];
                           [WORD ColumnWidth02];
                           [ENUM Column03];
                           [ANSISTRING ColumnFieldName03];
                           [WORD ColumnWidth03];
                           [ENUM Column04];
                           [ANSISTRING ColumnFieldName04];
                           [WORD ColumnWidth04];
                           [ENUM Column05];
```

```
[ANSISTRING ColumnFieldName05];
[WORD ColumnWidth05];
[ENUM Column06];
[ANSISTRING ColumnFieldName06];
[WORD ColumnWidth06];
[ENUM Column07];
[ANSISTRING ColumnFieldName07];
[WORD ColumnWidth07];
[ENUM Column08];
[ANSISTRING ColumnFieldName08];
[WORD ColumnWidth08];
[ENUM Column09];
[ANSISTRING ColumnFieldName09];
[WORD ColumnWidth09];
[ENUM Column10];
[ANSISTRING ColumnFieldName10];
[WORD ColumnWidth10];
[ENUM Column11];
[ANSISTRING ColumnFieldName11];
[WORD ColumnWidth11];
[ENUM Column12]
[ANSISTRING ColumnFieldName12];
[WORD ColumnWidth12];
[ENUM Column13];
[ANSISTRING ColumnFieldName13];
[WORD ColumnWidth13];
[ENUM Column14];
[ANSISTRING ColumnFieldName14];
[WORD ColumnWidth14];
[ENUM Column15];
[ANSISTRING ColumnFieldName15];
[WORD ColumnWidth15];
[ENUM Column16];
[ANSISTRING ColumnFieldName16];
[WORD ColumnWidth16];
[ENUM Column17];
[ANSISTRING ColumnFieldName17];
[WORD ColumnWidth17];
[ENUM Column18];
[ANSISTRING ColumnFieldName18];
[WORD ColumnWidth18];
[ENUM Column19];
[ANSISTRING ColumnFieldName19];
[WORD ColumnWidth19])
```

Parameters

FolderName As ANSISTRING

The name of the folder that the token operates on.

SettingsName As ANSISTRING

(Optional) Name of the set of display settings.

SettingsDesc As ANSISTRING

(Optional) Description of the set of display settings.

QuickViewerPerFolder As ENUM

(Optional) Specifies whether to display the folder in the QuickViewer.

0 No
1 Yes

ShowQuickViewer As ENUM

(Optional) Specifies whether to show the QuickViewer.

0 No
1 Yes

ViewMode As ENUM

(Optional) Specifies the view mode as follows:

1 Calendar
2 Conversation
3 Details
4 Checklist

ItemSourceIncoming As ENUM

(Optional)

0 No
1 Yes

ItemSourceOutgoing As ENUM

(Optional)

0 No
1 Yes

ItemSourcePersonal As ENUM

(Optional)

0 No
1 Yes

ItemSourceDraft As ENUM

(Optional)

0 No
1 Yes

ItemTypeAppointment As ENUM

(Optional)

0 No
1 Yes

ItemTypeDocument As ENUM

(Optional)

0 No
1 Yes

ItemTypeMail As ENUM

(Optional)

0 No
1 Yes

ItemTypeNote As ENUM

(Optional)

0 No
1 Yes

ItemTypePhone As ENUM

(Optional)

0 No
1 Yes

ItemTypeTask As ENUM

(Optional)

0 No
1 Yes

ContactTypePerson As ENUM

(Optional)

0 No
1 Yes

ContactTypeGroup As ENUM

(Optional)

0 No
1 Yes

ContactTypeResource As ENUM

(Optional)

0 No
1 Yes

ContactTypeOrganization As ENUM

(Optional)

0 No
1 Yes

FilterDaysForward As DWORD

(Optional)

FilterDaysBackward As DWORD

(Optional)

ShowGroupLabels As ENUM

(Optional)

0 No
1 Yes

NoColumnSummary As ENUM

(Optional)

0 No
1 Yes

HideNonChecklistItems As ENUM

(Optional)

0 No
1 Yes

SortKey As ENUM

(Optional) Specifies the sort key as follows:

600	AcceptedCnt
800	AdditionalRoute
112	AssignedDate
625	Author
4	Authority
126	BeginDateTime
847	Birthday
6	Caller
361	Category
802	CellPhoneNum
803	City
804	Comment
7	Company
603	CompletedCnt
825	ContactName
602	CopyType
805	Country
139	CreateDate
626	Creator
176	Date
606	DeletedCnt
605	DeletedDate
507	DeliveredDate
806	Department
656	DocCreatedDate

608	DocumentID
629	DocumentType
807	Domain
148	DueDate
808	EmailAddress
809	EMailType
151	EndDateTime
810	FaxPhoneNum
811	FirstName
1	From
812	Greeting
813	GUID
814	HomeAddress
815	HomeCity
816	HomeCountry
817	HomePhoneNum
818	HomeState
819	HomeZipCode
820	IDomain
801	IMScreenName
169	ItemType
821	LastName
822	LastReferenceDate
614	Library
647	LibraryId
823	Mailstop
824	MiddleName
633	Modified
184	Name
703	NamedFldByte
702	NamedFldDate
706	NamedFldSDWord
704	NamedFldSWord
700	NamedFldText
707	NamedFldUDWord

705	NamedFldUWord
701	NamedFldWrdStr
826	NDSDistinguishedName
827	NetworkID
828	OfficeAddress
829	OfficePhoneNum
830	OfficeWebsite
615	OpenedCnt
831	Organization
832	Owner
833	PagerPhoneNum
834	PersonalWebsite
8	Phone
835	PhoneNum
5	Place
836	PostOffice
837	Prefix
616	RecipientsCnt
838	ReferenceCount
617	RepliedCnt
657	SharerName
618	Size
619	Source
839	State
646	Subclass
9	Subject
227	TaskCategory
230	TaskPriority
351	ThreadStatus
841	Title
0	To
842	UserID
622	UndeliveredCnt
623	Version
634	VersionCreator

635	VersionDate
636	VersionDescription
267	ViewName
843	ZipCode

SortFieldName As ANSISTRING

(Optional) Name of the field to sort on.

SortOrder As ENUM

(Optional)

111 Ascending
146 Descending

UserID As ANSISTRING

(Optional) User ID of the mailbox where the display settings are being set.

Column00 As ENUM

(Optional) Use the same values as listed for SortKey.

ColumnFieldName00 As ANSISTRING

(Optional) Column name.

ColumnWidth00 As WORD

(Optional) Column width.

Column01 As ENUM

(Optional) Use the same values as listed for SortKey.

ColumnFieldName01 As ANSISTRING

(Optional) Column name.

ColumnWidth01 As WORD

(Optional) Column width.

Column02 As ENUM

(Optional) Use the same values as listed for SortKey.

ColumnFieldName02 As ANSISTRING

(Optional) Column name.

ColumnWidth02 As WORD

(Optional) Column width.

Column03 As ENUM

(Optional) Use the same values as listed for SortKey.

ColumnFieldName03 As ANSISTRING

(Optional) Column name.

ColumnWidth03 As WORD

(Optional) Column width.

Column04 As ENUM

(Optional) Use the same values as listed for SortKey.

ColumnFieldName04 As ANSISTRING

(Optional) Column name.

ColumnWidth04 As WORD

(Optional) Column width.

Column05 As ENUM

(Optional) Use the same values as listed for SortKey.

ColumnFieldName05 As ANSISTRING

(Optional) Column name.

ColumnWidth05 As WORD

(Optional) Column width.

Column06 As ENUM

(Optional) Use the same values as listed for SortKey.

ColumnFieldName06 As ANSISTRING

(Optional) Column name.

ColumnWidth06 As WORD

(Optional) Column width.

Column07 As ENUM

(Optional) Use the same values as listed for SortKey.

ColumnFieldName07 As ANSISTRING

(Optional) Column name.

ColumnWidth07 As WORD

(Optional) Column width.

Column08 As ENUM

(Optional) Use the same values as listed for SortKey.

ColumnFieldName08 As ANSISTRING

(Optional) Column name.

ColumnWidth08 As WORD

(Optional) Column width.

Column09 As ENUM

(Optional) Use the same values as listed for SortKey.

ColumnFieldName09 As ANSISTRING

(Optional) Column name.

ColumnWidth09 As WORD

(Optional) Column width.

Column10 As ENUM

(Optional) Use the same values as listed for SortKey.

ColumnFieldName10 As ANSISTRING

(Optional) Column name.

ColumnWidth10 As WORD

(Optional) Column width.

Column11 As ENUM

(Optional) Use the same values as listed for SortKey.

ColumnFieldName11 As ANSISTRING

(Optional) Column name.

ColumnWidth11 As WORD

(Optional) Column width.

Column12 As ENUM

(Optional) Use the same values as listed for SortKey.

ColumnFieldName12 As ANSISTRING

(Optional) Column name.

ColumnWidth12 As WORD

(Optional) Column width.

Column13 As ENUM

(Optional) Use the same values as listed for SortKey.

ColumnFieldName13 As ANSISTRING

(Optional) Column name.

ColumnWidth13 As WORD

(Optional) Column width.

Column14 As ENUM

(Optional) Use the same values as listed for SortKey.

ColumnFieldName14 As ANSISTRING

(Optional) Column name.

ColumnWidth14 As WORD

(Optional) Column width.

Column15 As ENUM

(Optional) Use the same values as listed for SortKey.

ColumnFieldName15 As ANSISTRING

(Optional) Column name.

ColumnWidth15 As WORD

(Optional) Column width.

Column16 As ENUM

(Optional) Use the same values as listed for SortKey.

ColumnFieldName16 As ANSISTRING

(Optional) Column name.

ColumnWidth16 As WORD

(Optional) Column width.

Column17 As ENUM

(Optional) Use the same values as listed for SortKey.

ColumnFieldName17 As ANSISTRING

(Optional) Column name.

ColumnWidth17 As WORD

(Optional) Column width.

Column18 As ENUM

(Optional) Use the same values as listed for SortKey.

ColumnFieldName18 As ANSISTRING

(Optional) Column name.

ColumnWidth18 As WORD

(Optional) Column width.

Column19 As ENUM

(Optional) Use the same values as listed for SortKey.

ColumnFieldName19 As ANSISTRING

(Optional) Column name.

ColumnWidth19 As WORD

(Optional) Column width.

DisplaySettingsSetEx4()

Customizes the display appearance of folder contents in the GroupWise client. See also DisplaySettingsSet(), DisplaySettingsSetEx(), DisplaySettingsSetEx2(), and DisplaySettingsSetEx3().

Token ID

```
AFTKN_DISPLAY_SETTINGS_SET_EX4 or 929
```

Syntax

```
VOID DisplaySettingsSetEx4(
     ANSISTRING FolderName;
     [ANSISTRING] SettingsName;
     [ANSISTRING] SettingsDesc;
     [ENUM] FolderListPerFolder;
     [ENUM] ShowFolderList;
     [ENUM] ShowFavoritesFolderList;
     [ENUM] ShowSimpleFolderList;
     [ENUM] ShowFullFolderList;
     [ENUM] QuickViewerPerFolder;
     [ENUM] ShowQuickViewer;
     [ENUM] ViewMode;
     [ENUM] ItemSourceIncoming;
     [ENUM] ItemSourceOutgoing;
     [ENUM] ItemSourcePersonal;
     [ENUM] ItemSourceDraft;
     [ENUM] ItemTypeAppointment;
     [ENUM] ItemTypeDocument;
     [ENUM] ItemTypeMail;
     [ENUM] ItemTypePhone;
     [ENUM] ItemTypeNote;
     [ENUM] ItemTypeTask;
     [ENUM] ContactTypePerson;
     [ENUM] ContactTypeGroup;
     [ENUM] ContactTypeResource;
     [ENUM] ContactTypeOrganization;
     [DWORD] FilterDaysForward;
     [DWORD] FilterDaysBackward;
     [ENUM] ShowGroupLables;
     [ENUM] NoColumnsSummary;
     [ENUM] MsgPreview;
     [ENUM] HideNonTasklistItems;
     [ENUM] HideCompletedItems;
     [DWORD] PanelColumnCount;
     [ENUM]SortKey;
     [ANSISTRING] SortFieldName;
     [ENUM] SortOrder;
     [ANSISTRING] UserID;
     [ENUM] Column00;
     [ANSISTRING] ColumnFieldName00;
     [Word] ColumnWidth00;
     [ENUM] Column01;
     [ANSISTRING] ColumnFieldName01;
     [Word] ColumnWidth01;
     [ENUM] Column02;
```

```
[ANSISTRING] ColumnFieldName02;
[Word] ColumnWidth02;
[ENUM] Column03;
[ANSISTRING] ColumnFieldName03;
[Word] ColumnWidth03;
[ENUM] Column04;
[ANSISTRING] ColumnFieldName04;
[Word] ColumnWidth04;
[ENUM] Column05;
[ANSISTRING] ColumnFieldName05;
[Word] ColumnWidth05;
[ENUM] Column06;
[ANSISTRING] ColumnFieldName06;
[Word] ColumnWidth06;
[ENUM] Column07;
[ANSISTRING] ColumnFieldName07;
[Word] ColumnWidth07;
[ENUM] Column08;
[ANSISTRING] ColumnFieldName08;
[Word] ColumnWidth08;
[ENUM] Column09;
[ANSISTRING] ColumnFieldName09;
[Word] ColumnWidth09;
[ENUM] Column10;
[ANSISTRING] ColumnFieldName10;
[Word] ColumnWidth10;
[ENUM] Column11;
[ANSISTRING] ColumnFieldName11;
[Word] ColumnWidth11;
[ENUM] Column12;
[ANSISTRING] ColumnFieldName12;
[Word] ColumnWidth12;
[ENUM] Column13;
[ANSISTRING] ColumnFieldName13;
[Word] ColumnWidth13;
[ENUM] Column14;
[ANSISTRING] ColumnFieldName14;
[Word] ColumnWidth14;
[ENUM] Column15;
[ANSISTRING] ColumnFieldName15;
[Word] ColumnWidth15;
[ENUM] Column16;
[ANSISTRING] ColumnFieldName16;
[Word] ColumnWidth16;
[ENUM] Column17;
[ANSISTRING] ColumnFieldName17;
[Word] ColumnWidth17;
[ENUM] Column18;
[ANSISTRING] ColumnFieldName18;
[Word] ColumnWidth18;
[ENUM] Column19;
[ANSISTRING] ColumnFieldName19;
[Word] ColumnWidth19)
```

Parameters

FolderListPerFolder As ENUM

(Optional)

0 No. This folder will use the global settings for displaying the folder list control.

1 Yes. This folder will use it's own settings for determining whether or not to show the folder control independent of the global settings.

ShowFolderList As ENUM

(Optional)

0 No, Hide the folder list.

1 Yes. Show the folder list.

ShowFavoritesFolderList As ENUM

(Optional)

0 No. Hide the favorites folder list.

1 Yes. Show the favorites folder list.

ShowSimpleFolderList As ENUM

(Optional)

0 No. Hide simple folder list.

1 Yes. Show the simple folder list.

ShowFullFolderList As ENUM

(Optional)

0 No. Hide the full folder list.

1 Yes. Show the full folder list.

ViewMode As ENUM

(Optional)

1 Calendar

2 Conversation

3 Details

4 Tasklist

5 Panels

6 AddressCard

MsgPreview As ENUM

(Optional)

0 No. Do not display the message preview.

1 Yes. Display a two line preview of the message body in the item list.

HideNonTasklistItems As ENUM

(Optional) This was renamed from HideNonChecklistItems in the DisplaySettingsSetEx3 token but is functionally equivalent.

0 No

1 Yes

HideCompletedItems As ENUM

(Optional)

0 No. Do not hide completed items.

370 Immediately. Hide completed items as soon as they are marked completed.

371 AfterADay. Hide completed items on the day following the day on which they were marked completed.

PanelColumnCount As DWORD

The number of columns to display in the panels view.

DisplaySettingsSetTemp()

Changes display settings temporarily for selected folder.

Token ID

```
AFTKN_DISPLAY_SETTINGS_SET_TEMP or 886
```

Syntax

```
VOID DisplaySettingsSetTemp(ANSISTRING SettingsName;
                            [ANSISTRING FolderName])
```

Parameters

SettingsName As ANSISTRING

Name of the set of display settings.

FolderName As ANSISTRING

(Optional). Defaults to currently selected folder.

DmDisplayErrors()

Displays DmDisplay errors.

Token ID

```
AFTKN_DM_DISPLAY_ERRORS or 882
```

Syntax

```
VOID DmDisplayErrors(ENUM ErrorDisplayState)
```

Parameters

ErrorDisplayState As ENUM

0 Off!
1 On!

FolderName As ANSISTRING

DocumentCheckInDlg()

Displays the Document Check-in dialog box.

Token ID

```
DTKN_DM_CHECKIN or 106
```

Syntax

```
VOID DocumentCheckInDlg()
```

See Also

“DocumentCheckOutDlg()” on page 239

DocumentCheckOutDlg()

Displays the Document Check-out dialog box which lists any document references selected in the GroupWise client.

Token ID

`DTKN_DM_CHECKOUT or 107`

Syntax

```
VOID DocumentCheckOutDlg()
```

See Also

“DocumentCheckInDlg()” on page 238

DocumentEndRetrieve()

Returns a document being edited to the GroupWise library and updates the profile and version information associated with the document.

Token ID

```
BFTKN_DM_ENDRETRIEVE or 348
```

Syntax

```
VOID DocumentEndRetrieve()
```

DocumentImportDlg()

Runs the Wizard to import new documents into the GroupWise library.

Token ID

```
DTKN_DM_IMPORT or 112
```

Syntax

```
VOID DocumentImportDlg()
```

DocumentMassOperationDlg()

Performs operations on multiple documents at once.

Token ID

```
DTKN_DM_MASS_OPERATION or 125
```

Syntax

```
VOID DocumentMassOperationDlg()
```

DocumentNewVersion()

Creates a new version of the document identified by the Document Number in DocIDStr. Depending on the value of Method; the user may be prompted for a new document version description.

Token ID

`AFTKN_DM_NEW_VERSION or 874`

Syntax

```
ANSISTRING DocumentNewVersion(ANSISTRING DocIDStr;
                              ENUM Method)
```

Parameters

DocIDStr As ANSISTRING

Document number of the document to create a new version for.

Method As ENUM

Specifies the method as follows:

200	Interactive Asks for folder to create reference and Version Desc dialog box.
201	NonInteractive

Return Values

ANSISTRING

Remarks

IMPORTANT: DocIDStr in the form Domain.PostOffice.Library:Doc#.Ver# for ObjectAPI reference: LibraryID: Doc#.Ver#. The DocIDStr is a subset of the object DocumentVersion.ODMADocumentID in the ObjectAPI.

DocumentNewVersionDlg()

Displays a dialog to create a new version of a document.

Token ID

```
DTKN_DM_NEW_VERSION or 118
```

Syntax

```
VOID DocumentNewVersionDlg()
```

DocumentReplace()

Replaces the document in the document library specified by the DocIDStr with the document specified with the file name.

Token ID

```
AFTKN_DM_REPLACE_DOC or 892
```

IMPORTANT: DocIDStr in the form Domain.PostOffice.Library:Doc#.Ver# for ObjectAPI reference: LibraryID: Doc#.Ver#. The DocIDStr is a subset of the object DocumentVersion.ODMADocumentID in the ObjectAPI.

Syntax

```
VOID DocumentReplace(ANSISTRING DocIDStr; ANSISTRING Filename)
```

Parameters

DocIDStr As ANSISTRING

Document number for the document to replace.

Filename As ANSISTRING

Name of the document to replace.

DocumentReplaceDlg()

Displays the Replace Document Version With Backup dialog box.

Token ID

```
DTKN_DM_REPLACE_DOC or 128
```

Syntax

```
VOID DocumentReplaceDlg()
```

DocumentSetInUseDlg()

Displays the Reset Document Status dialog which allows a user to specify documents to be marked as In Use in the GroupWise library.

Token ID

```
DTKN_DM_SETINUSE or 113
```

Syntax

```
VOID DocumentSetInUseDlg()
```

DocumentVersionList()

Lists a versions of the currently selected document.

Token ID

```
BFTKN_DM_VERSION_LIST or 350
```

Syntax

```
VOID DocumentVersionList()
```

E

This section contains information about the following tokens:

EchoDocument()

Echoes the document to the remote database.

Token ID

```
AFTKN_DM_ECHO_DOC or 884
```

Syntax

```
VOID EchoDocument(ANSISTRING DocIDStr)
```

Parameters

DocIDStr As ANSISTRING

In the form Domain.PostOffice.Library:Doc#.Ver#.

EditAttachmentToolbarDlg()

Adds, moves, or deletes buttons from the current Attachment Toolbar.

Token ID

```
DTKN_ATTBTNBAR_EDIT or 122
```

Syntax

```
VOID EditAttachmentToolbarDlg()
```

EditCopy()

Copies selected text to the Clipboard, replacing existing Clipboard text.

Token ID

```
BFTKN_COPY or 215
```

Syntax

```
VOID EditCopy()
```

See Also

“EditCut()” on page 254
“EditPaste()” on page 255
“OlePasteLink()” on page 616

EditCut()

Removes selected text and copies to the Clipboard, replacing existing Clipboard text.

Token ID

```
BFTKN_CUT or 214
```

Syntax

```
VOID EditCut()
```

See Also

“EditCopy()” on page 253
“EditPaste()” on page 255
“OlePasteLink()” on page 616

EditPaste()

Inserts Clipboard text at the insertion point.

Token ID

```
BFTKN_PASTE or 228
```

Syntax

```
VOID EditPaste()
```

See Also

“EditCopy()” on page 253
“EditCut()” on page 254
“OlePasteLink()” on page 616

Remarks

The message body field (in the Message View and the Object API) has a limit of 32,000 characters. The Message View will stop accepting data after 32,000 characters.

EditSettingsSelect()

Edit display settings for the current folder.

Token ID

```
DTKN_DISPLAY_SETTINGS_EDIT or 127
```

Syntax

```
VOID EditSettingsSelect([ANSISTRING FolderName])
```

Parameters

FolderName As ANSISTRING

(Optional) Folder name for editing the display settings.

EditToolbarDlg()

Brings up the edit toolbar dialog.

Token ID

```
DTKN_BTNBAR_EDIT or 38
```

Syntax

```
VOID EditToolbarDlg()
```

EmailImport()

Invokes the email import wizard for importing accounts and email from Outlook Express and Netscape.

Token ID

```
BFTKN_EMAIL_IMPORT or 1012
```

Syntax

```
VOID EmailImport()
```

EmptySelectedItems()

Removes selected items from Trash and erases them from the post office database.

Token ID

```
BFTKN_PURGE or 291
```

Syntax

```
VOID EmptySelectedItems()
```

EmptyTrash()

Removes all items from Trash and erases them from the post office database.

Token ID

```
BFTKN_EMPTY_TRASH or 207
```

Syntax

```
VOID EmptyTrash()
```

EndPreviewVersion()

Deletes the staged copy of the document specified by DocIDStr.

Token ID

```
AFTKN_DM_ENDPREVIEW or 856
```

Syntax

```
VOID EndPreviewVersion(ANSISTRING DocIDStr;
                       ANSISTRING PathFile)
```

Parameters

DocIDStr As ANSISTRING

Document number of the document to preview.

PathFile As ANSISTRING

In the form Domain.PostOffice.Library:Doc#.Ver#.

See Also

See DocumentVersion.Preview & EndPreview in the ObjectAPI documentation.

Enter()

Inserts a hard return at the insertion point, or opens a selected item.

Token ID

```
BFTKN_RETURN or 179
```

Syntax

```
VOID Enter()
```

Remarks

The message body field (in the Message View and the Object API) has a limit of 32,000 characters. The Message View will stop accepting data after 32,000 characters.

EnumSelect()

Selects an enumeration from a user-defined control such as a pop-up button.

Token ID

```
AFTKN_SET_ENUMVAL or 642
```

Syntax

```
VOID EnumSelect(ANSISTRING Text;
                  [ANSISTRING ControlName])
```

Parameters

Text As ANSISTRING

Enumeration to select, such as a list item on a pop-up button control.

ControlName As ANSISTRING

(Optional) Name of a user-defined control. For example, open GroupWise View Designer and create a pop-up button control. Then choose Edit | Item Properties. Type the ControlName in the List Name edit box. This is case sensitive with the actual name given to the control in the designer.

EnvCheckCurrentWindow()

Returns True if the current window type matches a specified window type; False, if not.

Token ID

```
AFTKN_CURRENT_WINDOW or 662
```

Syntax

```
BOOLEAN EnvCheckCurrentWindow(ENUM WindowType)
```

Parameters

WindowType As ENUM

Specifies the window type as follows:

109	AppointmentMessage
118	AttachmentView
624	Browser
1	Calendar
6	Inbox
168	Info
304	ItemView
175	MailMessage
177	MainWindow
181	ModelessDialog
189	NoteMessage
7	Outbox
288	PhoneMessage
229	TaskMessage
8	Trash

Return Values

BOOLEAN

True or False

EnvClearChangedFlag()

Resets the view-has-been-modified flag on all view controls to zero, which disables the "Do you want to save your changes?" dialog when closing the view.

Token ID

```
BFTKN_ENV_CLEAR_CHANGED_FLAG or 205
```

Syntax

```
VOID EnvClearChangedFlag()
```

EnvClipboardText()

Returns the contents (text) of the clipboard.

Token ID

```
AFTKN_CLIPBOARD_TEXT or 737
```

Syntax

```
ANSISTRING EnvClipboardText()
```

Return Values

ANSISTRING.

Clipboard text.

EnvCommandLine()

Returns the command line string passed to GroupWise at startup.

Token ID

```
AFTKN_COMMAND_LINE or 770
```

Syntax

```
ANSISTRING EnvCommandLine()
```

Return Values

ANSISTRING

Command line string.

EnvCurrentViewName()

Returns the name of the active view.

Token ID

```
AFTKN_ENV_CURRENTVIEWNAME or 784
```

Syntax

```
ANSISTRING EnvCurrentViewName()
```

Return Values

ANSISTRING

Name of active view.

EnvDefaultConnectionName()

Returns the name of the default connection, which is a collection of settings used to hook up to the master mailbox that includes IP address and port for TCP/IP connections, etc.

Token ID

```
AFTKN_REM_GETDEFAULTCONNECTNAME or 798
```

Syntax

```
ANSISTRING EnvDefaultConnectionName()
```

Return Values

ANSISTRING.

EnvDocumentGetIntegrations()

Determines if Document Management Integrations is enabled.

Token ID

```
AFTKN_DM_GET_INTEGRATIONS_SWITCH or 879
```

Syntax

```
BOOLEAN EnvDocumentGetIntegrations()
```

Return Values

BOOLEAN. Returns True if Document Integrations is enabled. Returns False otherwise.

EnvEditorStyle()

Determines if HTMLview editor is enabled.

Token ID

`AFTKN_MSG_EDITOR_STYLE or 894`

Syntax

```
WORD EnvEditorStyle()
```

Return Values

WORD. Only returns valid information if a compose mail view is present. Returns 2 if the active compose mail view is using the HTML editor, and 1 otherwise.

EnvGetNumLibraries()

Returns the number of GroupWise libraries on the users' post office.

Token ID

```
AFTKN_DM_GET_NUM_LIBRARIES or 881
```

Syntax

```
WORD EnvGetNumLibraries()
```

Return Values

WORD

EnvGetProfileField()

Returns the value of the field specified by FieldID in the document profile for the document specified by the Document Number in DocIDStr.

Token ID

```
AFTKN_DM_GETFIELD or 862
```

Syntax

```
ANSISTRING EnvGetProfileField(ANSISTRING DocIDStr;
                                  ENUM FieldID)
```

Parameters

DocIDStr As ANSISTRING

Document number of the document.

FieldID As ENUM

Specifies the field as follows:

625	Author
628	Document Extension
	Returns the Document Extension of the file name.
629	Document Type
9	Subject

Return Values

ANSISTRING

Field String.

EnvIsCommandValid()

Checks to see if command is valid.

Token ID

```
AFTKN_ENV_ISCOMMANDVALID or 783
```

Syntax

```
BOOLEAN EnvIsCommandValid(WORD CommandID)
```

Parameters

CommandID As WORD

Name of command.

Return Values

BOOLEAN. Returns True if the token indicated by CommandID is valid given the current state of GroupWise. Returns False otherwise.

EnvIsNetworkUser()

Returns True if the current user is logged into the network; False, if not. This command detects whether the /la startup option is used.

Token ID

```
AFTKN_IS_NETWORK_USER or 772
```

Syntax

```
BOOLEAN EnvIsNetworkUser()
```

Return Values

BOOLEAN

True/False.

EnvIsRemote()

Returns True if GroupWise is running as a remote client; False, if not.

Token ID

`AFTKN_IS_REMOTE or 738`

Syntax

```
BOOLEAN EnvIsRemote()
```

Return Values

BOOLEAN

True/False.

EnvIsRemoteConnectionActive()

Returns True if the remote connection is active; False, if not.

Token ID

```
AFTKN_ISREMOTECONNECTED or 777
```

Syntax

```
BOOLEAN EnvIsRemoteConnectionActive()
```

Return Values

BOOLEAN

True/False.

EnvIsRemoteConnectionFinished()

Returns True if the updating of the remote mailbox is complete and there are no outstanding remote requests. False if not.

Token ID

```
AFTKN_ISREMCONNECTDONE or 779
```

Syntax

```
BOOLEAN EnvIsRemoteConnectionFinished()
```

Return Values

BOOLEAN

True/False.

EnvIsRemoteConnectionNeeded()

Returns True if a remote connection is needed; False, if not. A connection is needed when items are in the Pending Requests to Master Mailbox dialog box. Open GroupWise, choose Remote > Pending Requests.

Token ID

```
AFTKN_ISREMCONNECTNEEDED or 778
```

Syntax

```
BOOLEAN EnvIsRemoteConnectionNeeded()
```

Return Values

BOOLEAN

True/False.

EnvLastCmdResult()

Returns the value returned by the last DDE-issued command.

Token ID

```
AFTKN_RETURNVAL_GET or 760
```

Syntax

```
ANSISTRING EnvLastCmdResult()
```

Return Values

ANSISTRING

Value returned by the last DDE-issued command.

EnvLastError()

Returns information about an error condition that causes a DDE or other transaction to fail.

Token ID

```
AFTKN_LASTERROR or 765
```

Syntax

```
ANSISTRING EnvLastError()
```

Return Values

ANSISTRING

Error condition, as follows.

10E1	Bad Parameter x (x is the index of a bad parameter)
10E2	Failed
10E3	Command is invalid in the current GroupWise state
10E4	Not found
10E5	Syntax error
10E7	BC does not apply to personal items
10E8	Not supported for encapsulated item attachments
10E9	Not supported for OLE attachments
10EA	Message ID references a Sent Items message that does not exist yet
10EB	You cannot set an alarm for a time that has already expired
10EC	The Mailbox and Sent Items source attributes of the Mailbox filter cannot be changed
10ED	The source attributes of the Sent Items filter cannot be changed

Remarks

Use this command as part of a DDE request transaction. Possible error strings are:

EnvLastWindowCanceled()

Returns True if the last displayed window was canceled; False, if not.

Token ID

```
AFTKN_LAST_WINDOW_CANCELED or 771
```

Syntax

```
BOOLEAN EnvLastWindowCanceled()
```

Return Values

BOOLEAN

True/False.

EnvNetworkLoginID()

Returns the network login ID of the current user.

Token ID

```
AFTKN_NETWORK_LOGIN_ID or 739
```

Syntax

```
ANSISTRING EnvNetworkLoginID()
```

Return Values

ANSISTRING

Network login ID.

See Also

“EnvIsNetworkUser()” on page 275

EnvPrefArchivePath()

Returns the default path of archive database files.

Token ID

```
AFTKN_PREF_ARCHIVE_PATH or 741
```

Syntax

```
ANSISTRING EnvPrefArchivePath()
```

Return Values

ANSISTRING

Archive directory path.

EnvPrefCustomViewPath()

Returns the default path of custom view files.

Token ID

```
AFTKN_PREF_CUSTOM_VIEW_PATH or 742
```

Syntax

```
ANSISTRING EnvPrefCustomViewPath()
```

Return Values

ANSISTRING

Custom view directory path.

EnvPrefFullName()

Returns a user’s full name.

Token ID

```
AFTKN_PREF_FULL_NAME or 743
```

Syntax

```
ANSISTRING EnvPrefFullName()
```

Return Values

ANSISTRING

User’s full name.

EnvPrefPostOfficePath()

Returns the default post office path.

Token ID

```
AFTKN_PREF_POST_OFFICE_PATH or 745
```

Syntax

```
ANSISTRING EnvPrefPostOfficePath()
```

Return Values

ANSISTRING

Post office directory path.

EnvPrefSavePath()

Returns the default path for item (*.MLM) files and attachments.

Token ID

```
AFTKN_PREF_SAVE_PATH or 746
```

Syntax

```
ANSISTRING EnvPrefSavePath()
```

Return Values

ANSISTRING

Item and Attachment save directory path.

EnvSentMessageID()

Returns actual messageID of a new message created by the Sendxxx() tokens.

Token ID

```
AFTKN_ITEM_GET_SENT_MSGID or 830
```

Syntax

```
ANSISTRING EnvSentMessageID(DWORD IDFromSendToken;
                                [ENUM RemoveFromQueue])
```

Parameters

IDFromSendToken As DWORD

Description of argument.

RemoveFromQueue As ENUM

(Optional) Enumerated Values:

0 No
1 Yes

Return Values

MessageId of new message as an ANSISTRING.

EnvTextCurrentCharIndex()

Returns the position of the cursor in the current message.

Token ID

```
AFTKN_TEXTENTRY_GET_CHAR_INDEX or 659
```

Syntax

```
WORD EnvTextCurrentCharIndex()
```

Return Values

WORD.

Remarks

Works in both compose and read modes for RTF, but only in compose mode for HTML. Recommend use in RTF.

EnvTextCurrentLineIndex()

Identifies the current curser location by line number.

Token ID

```
AFTKN_TEXTENTRY_GET_LINE_INDEX or 658
```

Syntax

```
WORD EnvTextCurrentLineIndex()
```

Return Values

WORD. Returns the line number associated with the current cursor position.

Remarks

This token uses zero-based indexes, thus the first line of a message is line 0, the next link is line 1, etc.

EnvTextCurrentWord()

Returns the word at the current cursor position.

Token ID

```
AFTKN_TEXTENTRY_GET_CURWORD or 660
```

Syntax

```
ANSISTRING EnvTextCurrentWord()
```

Return Values

ANSISTRING.

EnvTextInsertMode()

Detects the insert mode of editor.

Token ID

```
AFTKN_TEXTENTRY_IS_INSERT_MODE or 813
```

Syntax

```
BOOLEAN EnvTextInsertMode()
```

Return Values

BOOLEAN. Returns True if the message editor is currently in insert mode, False otherwise.

EnvTextLeftChar()

Returns the character to the left of the cursor in the message window.

Token ID

```
AFTKN_TEXTENTRY_GET_LEFTCHAR or 661
```

Syntax

```
ANSISTRING EnvTextLeftChar()
```

Return Values

ANSISTRING.

EnvTextLineCharCount()

Returns a number of characters on the current line in the message window.

Token ID

```
AFTKN_TEXTENTRY_LINE_CHARCOUNT or 656
```

Syntax

```
WORD EnvTextLineCharCount(WORD Index)
```

Parameters

Index As WORD

(Optional)

Return Values

WORD

EnvTextLineCount()

Returns the number of lines in a current message box (must have input focus); 1, if empty.

Token ID

```
AFTKN_TEXTENTRY_GET_LINE_COUNT or 657
```

Syntax

```
WORD EnvTextLineCount()
```

Return Values

WORD

Number of lines.

Remarks

The message body field (in the Message View and the Object API) has a limit of 32,000 characters. The Message View will stop accepting data after 32,000 characters.

EnvTextRightChar()

Returns the character to the right of the cursor in the message window.

Token ID

```
AFTKN_TEXTENTRY_GET_RIGHTCHAR or 795
```

Syntax

```
ANSISTRING EnvTextRightChar()
```

Return Values

ANSISTRING.

EnvUserID()

Returns the current user ID.

Token ID

```
AFTKN_USERID or 764
```

Syntax

```
ANSISTRING EnvUserID([ENUM UserType];
                        [ENUM IDStyle])
```

Parameters

UserType As ENUM

(Optional) Enumerated Values:

1 ActiveUser
0 RootUser

IDStyle As ENUM

(Optional) Enumerated Values:

1 Full
0 Partial
2 InternetEmail - guaranteed to return the email address in Internet format (user@domain)

Return Values

ANSISTRING UserID

EnvVersionDate()

Returns the GroupWise version date in form: MM/DD/YY.

Token ID

```
AFTKN_VERSION_DATE or 749
```

Syntax

```
ANSISTRING EnvVersionDate()
```

Return Values

ANSISTRING

GroupWise version date.

EnvVersionName()

Returns the GroupWise version name.

Token ID

```
AFTKN_VERSION_NAME or 748
```

Syntax

```
ANSISTRING EnvVersionName()
```

Return Values

ANSISTRING

GroupWise version name.

EventNoticeRegister()

Returns a handle.

Token ID

```
AFTKN_EN_REGISTER or 815
```

Syntax

```
DWORD EventNoticeRegister(ANSISTRING DLLPath;
                          ANSISTRING FunctionName;
                          WORD RegisteredID;
                              WORD Flags)
```

Parameters

DLLPath As ANSISTRING

Text. Fully qualified file name to a dll/exe containing the function to call when an event occurs.

FunctionName As ANSISTRING

Text. Name of the exported function to call.

RegisteredID As WORD

A 32-bit ID value that will be passed to the function specified in DLLPath and FunctionName. This provides some flexibility since a pointer or handle can be passed to the memory where the function is handled.

Flags As WORD

The events you want notification for. The following are valid events:

0x00000001 EVT_GROUPWISE_SEND_ITEM
0x00000002 EVT_GROUPWISE_CANCEL_ITEM
0x00000004 EVT_GROUPWISE_CLOSE_ITEM
0x00000008 EVT_GROUPWISE_PACKAGING_ITEM
0x00000010 EVT_GROUPWISE_PACKAGING_ITEM_CANCELED
0x00000020 EVT_GROUPWISE_SEND_ITEM_COMPLETED

Return Values

DWORD

Handle.

Remarks

The first important area of code is in Command Execute within the C3PO. GWCommander is used to throw EventNoticeRegister. EventNoticeRegister sets up this association only for the active compose view. If the dialog is not up and focused, the token fails. If the active view is not a compose view, the token fails.

See Also

ItemGetText() (page 487)

EventNoticeRegisterEx()

Registers a third party handler to receive callbacks when the user takes an action on an open compose view that would result in the message being sent, saved, or closed. Note that this function is different from EventNoticeRegister() in the callback function must support a third parameter which after a successful send will contain the message ID of the message in the database.

Token ID

```
AFTKN_EN_REGISTER_EX or 944
```

Syntax

```
DWORD EventNoticeRegisterEX(
    ANSISTRING DLLPath;
    ANSISTRING FunctionName;
    DWORD RegisteredID;
    DWORD Flags)
```

Parameters

DLLPath As ANSISTRING

Text. Fully qualified file name to a dll/exe containing the function to call when an event occurs.

FunctionName As ANSISTRING

Text. Name of the exported function to call. The function should adhere to the following specifications:

HRESULT_stdcall (DWORD dwId, DWORD dwNotifyType, char *pMsgId);

- dwId - The “RegisteredID” that is passed into EventNoticeRegisterEx().
- dwNotifyType - Will be one of the following:

  0x00000001 EVT_GROUPWISE_SEND_ITEM
  0x00000002 EVT_GROUPWISE_CANCEL_ITEM
  0x00000004 EVT_GROUPWISE_CLOSE_ITEM
  0x00000008 EVT_GROUPWISE_PACKAGING_ITEM
  0x00000010 EVT_GROUPWISE_PACKAGING_ITEM_CANCELED
  0x00000020 EVT_GROUPWISE_SEND_ITEM_COMPLETED
- pMsgId - The GroupWise message ID of the message that was just created. This parameter is only non-null for the EVT_GROUPWISE_SEND_ITEM_COMPLETED notification.

RegisteredID As WORD

A 32-bit ID value that will be passed to the function specified in DLLPath and FunctionName. This provides some flexibility since a pointer or handle can be passed to the memory where the function is handled.

Flags As WORD

The events you want notification for. The following are valid events:

0x00000001 EVT_GROUPWISE_SEND_ITEM
0x00000002 EVT_GROUPWISE_CANCEL_ITEM
0x00000004 EVT_GROUPWISE_CLOSE_ITEM

0x00000008 EVT_GROUPWISE_PACKAGING_ITEM
0x00000010 EVT_GROUPWISE_PACKAGING_ITEM_CANCELED
0x00000020 EVT_GROUPWISE_SEND_ITEM_COMPLETED

Return Values

Returns a DWORD containing a handle that must be passed to EventNoticeUnregistered() in order to free up memory that was allocated for the event tracking.

EventNoticeUnregister()

Unregisters a handle.

Token ID

```
AFTKN_EN_UNREGISTER or 816
```

Syntax

```
VOID EventNoticeUnregister(DWORD Handle)
```

Parameters

Handle As DWORD

EventNotify()

Signals a third-party DLL when certain events happen. The method that this token calls can only be loaded from a DLL. Do not publish this token from your DLLs.

Token ID

```
AFTKN_EVENT_NOTIFY or 805
```

Syntax

```
VOID EventNotify(ENUM AppEvent;
                 ANSISTRING Command)
```

Parameters

AppEvent As ENUM

Specifies the application event as follows:

293	AppInitialized Indicates GroupWise is loaded and running. The command string is set to GROUPWISE: App Initialized.
294	BadCustomCommand
295	BadCustomMessage
296	ContextMenu Indicates context-sensitive QuickMenu display on right click. The command string contains an ANSISTRING menu identifier representation and the menu handle, both space delimited.

Command As ANSISTRING

If AppEvent equals AppInitialized, then command equals "WPOF: App Initialized."

If AppEvent equals ContextMenu, then command is a string representation of the decimal ID (MENU_ID) followed by a space and the decimal equivalent of the menu handle (HMENU) for the context menu that is about to be displayed. This allows the third-party DLL to add menu items to the context menu before it is displayed. EventNotify supports the following menu IDs:20 OLE Object21 Shelf Icon or Quick Start Icon22 Attachment control (composing message; attachment selected)23 Attachment control (reading message)

Remarks

The first event signals a third-party DLL that GroupWise is running. The second and third events signal that a custom command or message failed. The fourth event signals that a context menu (right mouse click menu) is about to be displayed. See AppEvent parameter.

Intercept the EventNotify() token and monitor the ContextMenu! case to add GroupWise QuickMenu items. Menu identifiers and their activation contexts are shown below:

MENU_ID and QuickMenu Context

20 OLE Object

21 Shelf Icon or Quick Start Icon
22 Attachment Control (composing message, attachment selected)
23 Attachment Control (reading message)
24 Sound Item or Control
25 Item List Control (In Box, Out Box, Calendar Views)
26 Movie Control
27 Reserved
28 Main Window Icons (In Box, Out Box, Calendar, Mail, Appointment, and so forth)
29 In Box View
30 Out Box View
31 Trash View
32 Reserved
33 Reserved
34 Reserved
35 Item List (preempted by local menu ID 25, but can be accessed in the control space to the left of the Column Manager)
36 Basic View (might be overridden by more specific view menus)
37 Basic Item View (might be overridden by more specific view menus)
38 Folder List Control (In Box, Out Box, Trash, Calendar Views)
39 Button Bar
40 CC: and BC: Controls (composing message)
41 Main Windows Shelf (no Shelf Icon selected)
42 Attachment Control (composing message, no attachment selected)
43 Trash's Item List (preempted by local menu ID 31 but can be accessed in the control space to the left of the column Manager)
44 Reserved
45 Reserved
46 Start Date/Auto-Date Control (composing message)
47 End Date/Duration Control (composing message)
48 To: Control (composing message)
49 Main Window's Trash Icon.

ExpandAll()

If the folder tree has focus, expand either the currently selected branch or the optionally specified branch of the folder tree such that all of it's children are displayed. If the item list has focus, expand all entries in a hierarchically displayed list such that all entries (including non-root items) are showing. This applies to lists viewed by discussion thread and tasklists that contain subtasks.

Token ID

```
BFTKN_EXPAND_ALL or 1045
```

Syntax

```
VOID ExpandAll(
      [ANSISTRING] FolderName)
```

Parameters

FolderName As ANSISTRING

(Optional) The full path of the folder in the folder tree whose children should be collapsed. Note that this parameter is ignored if the token is thrown at the item list.

ExpandFolders()

Expands the currently selected folder.

Token ID

```
BFTKN_CONTROL_PLUS 371
```

Syntax

```
void ExpandFolders()
```

ExpandThread()

Unregisters a handle.

Token ID

```
BFTKN_EXPAND_THREAD or 392
```

Syntax

```
VOID ExpandThread()
```

ExportContact()

The user is prompted for information on which contacts to export, where to save the file and what type of export format to use.

Token ID

```
BFTKN_EXPORT_CONTACT or 1247
```

Syntax

```
VOID ExportContact()
```

ExportContactBook()

Export the entire contents of the currently selected contact folder.

Token ID

```
BFTKN_EXPORT_CONTACT_BOOK or 1283
```

Syntax

```
VOID ExportContactBook()
```

Remarks

This token is only valid when a contacts folder is selected.

ExportSelectedContact()

Export the currently selected item or items in the currently displaying contacts folder.

Token ID

```
BFTKN_EXPORT_CONTACT_SELECTED or 12849
```

Syntax

```
VOID ExportSelectedContact()
```

F

This section contains information about the following tokens:

FavoritesFolderList()

Toggle the display of the favorites folder list.

Token ID

```
BFTKN_FOLDER_TREE_FAVORITES or 1278
```

Syntax

```
VOID FavoritesFolderList()
```

FilterApply()

Applies a filter to a specified list or control.

Token ID

```
AFTKN_FILTER_APPLY or 721
```

Syntax

```
VOID FilterApply(ENUM FilterHandleType;
                 ENUM ApplyTo;
                 [ANSISTRING ControlName])
```

Parameters

FilterHandleType As ENUM

Unique filter identifier, returned by FilterCreate():

315 Default

ApplyTo As ENUM

Specifies the item to apply the filter to as follows:

108	AppointmentList
44	ItemList
	Includes all items.
188	NoteList
219	SelectedControl
228	TaskList
266	View
268	WeekControl
	Includes Week calendar view items.

ControlName As ANSISTRING

(Optional) Name of user-defined control.

See Also

“FilterCreate()” on page 321

FilterCategory()

Filters the current list of messages by category.

Token ID

```
AFTKN_FILTER_CATEGORY 899
```

Syntax

```
void FilterCategory(ENUM FilterAction;
                    [ANSISTRING CategoryName])
```

Parameters

FilterAction as ENUM

Specifies the following:

844 Clear
845 Set
846 Prompt

CategoryName as ANSISTRING

(Optional) Name of the category with which to filter (for FilterAction Set!).

FilterCategorySelected()

Filters the item list using the category of the selected item or items as the filter criteria.

Token ID

`BFTKN_FILTER_CAT_SELECTED or 1223`

Syntax

```
void FilterCategorySelected()
```

Remarks

If the selected item in the mailbox item list has the “Follow Up” category, then issuing this token will filter the mailbox to only show items with the “Follow Up” category. If an item has multiple categories then it will filter item that have any of the matching categories. Similarly, if multiple items are selected, then the filter contains all the superset of categories on all of the selected items.

Note that at least one item must be selected in the list for this token to be enabled.

FilterClear()

Clears the contents of a filter.

Token ID

```
BFTKN_FILTER_CLEAR or 355
```

Syntax

```
VOID FilterClear()
```

See Also

“FilterCreate()” on page 321

FilterCreate()

Creates a new filter.

Token ID

```
AFTKN_FILTER_CREATE or 722
```

Syntax

```
DWORD FilterCreate(ENUM FilterType;
                   BOOLEAN DefaultFilter)
```

Parameters

FilterType As ENUM

Specifies the filter type as follows:

1	Calendar
150	Empty
6	Inbox
7	Outbox

Calendar!, In Box!, and Out Box!

Automatically applies defined filter criteria. For example, since Calendar! only displays appointments, notes, and tasks, you do not have to filter mail items (see FilterSetItemType()). Empty! requires you to set all filter criteria, including type, source, and attributes as needed.

DefaultFilter As BOOLEAN

Return Values

DWORD

Filter handle (a number greater than zero, or zero if an error occurs).

See Also

“FilterApply()” on page 317
“FilterClear()” on page 320

Remarks

IMPORTANT: Delete the filter handle when it is no longer needed, or the system resources will be lost. See “FilterDelete()” on page 322.

FilterDelete()

Deletes a filter handle from memory

Token ID

```
AFTKN_FILTER_DELETE or 723
```

Syntax

```
VOID FilterDelete(ENUM FilterHandleType)
```

Parameters

FilterHandleType As ENUM

Unique filter identifier:

315 Default

Remarks

IMPORTANT: Delete the filter handle when it is no longer needed, or the system resources will be lost. See “FilterCreate()” on page 321.

FilterDlg()

Displays the Filter, Filter Mailbox, or Filter Sent Items dialog box, depending on the view displayed.

Token ID

```
DTKN_FILTER_ITEMLIST or 58
```

Syntax

```
VOID FilterDlg()
```

FilterFromFile()

Retrieves filter information saved to a file.

Token ID

```
AFTKN_FILTER_FROM_FILE or 586
```

Syntax

```
VOID FilterFromFile(ANSISTRING Filename)
```

Parameters

Filename As ANSISTRING

Name of filter information file.

FilterGroupBegin()

Encapsulates a set of filter comparison operations to be evaluated together as a group, when paired with FilterGroupEnd(). Call FilterGroupMarker() to identify the group, then call FilterGroupBegin to start a group. Call the desired filter tokens to build up the group, and then call FilterGroupEnd to terminate the group.

Token ID

```
AFTKN_FILTER_GROUP_BEGIN or 840
```

Syntax

```
VOID FilterGroupBegin(ENUM FilterHandleType;
                      ENUM FilterOperator)
```

Parameters

FilterHandleType As ENUM

Unique filter identifier:

315 Default

FilterOperator As ENUM

316 And
317 Or

FilterGroupMarker()

Inserts a marker in a filter expression and operates on the grouped set of expressions that follow and precede it. Call FilterGroupBegin() to start a group. Call the desired filter tokens to build up the group, and then call FilterGroupEnd() to terminate the group.

Token ID

```
AFTKN_FILTER_GROUP_MARKER or 842
```

Syntax

```
VOID FilterGroupMarker(ENUM FilterHandleType;
                       ENUM FilterOperator)
```

Parameters

FilterHandleType As ENUM

Unique filter identifier:

315 Default

FilterOperator As ENUM

316 And
317 Or

Remarks

Rather than actually identifying a group, FilterGroupMarker operates on the groups before and after. For example, consider the following code

```
Commandr.Execute('FilterGroupBegin('+Filter+'Or!)' ,RetString);

Commandr.Execute('FilterGroupBegin('+Filter+'And!)' ,RetString);
Commandr.Execute('filtersetDate('+Filter+'CreateDate!;;;GTE!;Yesterday!;-15)'
     ,RetString);
Commandr.Execute('FilterGroupEnd('+Filter+')' ,RetString);

Commandr.Execute('FilterGroupMarker('+Filter+'Or!)' ,RetString);

Commandr.Execute('FilterGroupBegin('+Filter+'And!)' ,RetString);
Commandr.Execute('filtersetDate('+Filter+'CreateDate!;;;Equal!;Today!;)'
     ,RetString);
Commandr.Execute('FilterGroupEnd('+Filter+')' ,RetString);

Commandr.Execute('FilterGroupEnd('+Filter+')' ,RetString);
```

`FilterGroupMarker` will 'OR' the results of the grouped set before and the grouped set after. If you viewed this filter through the client, it would look like the following

```
(Created is greater than or equal to 15 Days before Yesterday) OR (Created is
Today)
```

FilterGroupEnd()

Encapsulates a set of filter comparison operations to be evaluated together as a group, when paired with FilterGroupBegin(). Call FilterGroupMarker() to identify the group, then call FilterGroupBegin to start a group. Call the desired filter tokens to build up the group, and then call FilterGroupEnd to terminate the group.

Token ID

`AFTKN_FILTER_GROUP_END or 841`

Syntax

```
VOID FilterGroupEnd(ENUM FilterHandleType)
```

Parameters

FilterHandleType As ENUM

Unique filter identifier:

315 Default

FilterMenuMore()

Displays a list of all saved filters.

Token ID

```
BFTKN_FILTER_MENU_MORE or 472
```

Syntax

```
VOID FilterMenuMore()
```

FilterReset()

Selects all the items in the filter results list.

Token ID

```
BFTKN_SELECT_ALL or 376
```

Syntax

```
VOID FilterReset()
```

FilterSetAttachmentClass()

Filters items according to the contents of the Attach list box.

Token ID

```
AFTKN_FILTER_SET_ATTACH_CLASS or 724
```

Syntax

```
VOID FilterSetAttachmentClass(ENUM FilterHandleType;
                              ENUM AttachmentClassType;
                              ENUM BitOn)
```

Parameters

FilterHandleType As ENUM

Unique filter identifier, returned by FilterCreate():

315 Default

AttachmentClassType As ENUM

Attachment type to filter, or not filter, depending on the BitOn parameter setting. Use multiple commands to filter more than one attachment type. Attachment class types include:

114	AttachClassFile
115	AttachClassMessage
116	AttachClassMovie
645	AttachClassOLE
117	AttachClassSound

BitOn As ENUM

Specifies how to filter the item type, as follows:

147	DontCare
0	No
	Filters out the specified item type.
1	Yes
	Passes the specified item type through the filter, but no others.

FilterSetAttribute()

Filters items by specified attributes.

Token ID

```
AFTKN_FILTER_SET_ATTRIBUTE or 725
```

Syntax

```
VOID FilterSetAttribute(ENUM FilterHandleType;
                        ENUM Attribute;
                        ENUM IsRequired)
```

Parameters

FilterHandleType As ENUM

Unique filter identifier, returned by FilterCreate():

315 Default

Attribute As ENUM

Attribute to filter, or not filter, depending on the IsRequired parameter setting. Use multiple commands to filter more than one attribute type. Attributes include:

104	Accepted
279	Archived
289	Autodated
138	Completed
141	CustomView
143	Delegated
144	Deleted
280	Opened
198	Personal
200	Private
206	Read
207	ReplyRequested
215	ReturnReceiptRequested
217	Routed

IsRequired As ENUM

147 DontCare!

0 No!
1 Yes!

FilterSetByte()

Filters items which have the user-defined field FieldName associated with it, based on the FieldOperator and ByteValue specified.

Token ID

```
AFTKN_FILTER_SET_BYTE or 837
```

Syntax

```
VOID FilterSetByte(ENUM FilterHandleType;
                   ANSISTRING FieldName;
                   ENUM FieldOperator;
                   WORD ByteValue;
                   [ANSISTRING UserID])
```

Parameters

FilterHandleType As ENUM

Unique filter identifier, returned by FilterCreate():

315 Default

FieldName As ANSISTRING

FieldOperator As ENUM

Specifies the field operator to use.

147	DontCare
320	Equal
321	GT
322	GTE
318	LT
319	LTE
323	NotEqual

ByteValue As WORD

UserID As ANSISTRING

(Optional)

FilterSetDate()

Filters items based on the item date attribute specified by FilterDateType, the date value specified by Date, and the operation specified by Operation. Optionally, the filter can specify a user-defined field with type Date by setting the optional FieldName parameter. In this case the filter uses this field with Date and Operation to filter for items.

Token ID

```
AFTKN_FILTER_SET_DATE or 845
```

Syntax

```
VOID FilterSetDate(ENUM FilterHandleType;
                   ENUM FilterDateType;
                   [ANSISTRING FieldName];
                   [ANSISTRING UserID];
                   ENUM Operation;
                   ENUM Date;
                   [DWORD DateOffset])
```

Parameters

FilterHandleType As ENUM

Unique filter identifier, returned by FilterCreate():

315 Default

FilterDateType As ENUM

Specifies the filter date type as follows:

112	AssignedDate
126	BeginDateTime
139	CreateDate
507	DeliveredDate
148	DueDate
151	EndDateTime
331	NamedField

FieldName As ANSISTRING

(Optional)

UserID As ANSISTRING

(Optional)

Operation As ENUM

Specifies the operation as follows:

320	Equal
321	GT
322	GTE
318	LT
319	LTE

Date As ENUM

Specifies the date as follows:

3	Month
235	Today
236	Tomorrow
2	Week
4	Year
271	Yesterday

DateOffset As DWORD

(Optional)

FilterSetDateAbsolute()

Filters items according to an absolute date or date range.

Token ID

```
AFTKN_FILTER_SET_DATE_ABSOLUTE or 726
```

Syntax

```
VOID FilterSetDateAbsolute (ENUM FilterHandleType;
                            ENUM FilterDateType;
                           [ANSISTRING FieldName];
                           [ANSISTRING UserID];
                            ENUM Operation;
                            WORD DayOfMonth;
                            WORD Month;
                            WORD Year])
```

Parameters

FilterHandleType As ENUM

Unique filter identifier, returned by FilterCreate():

315 Default

FilterDateType As ENUM

Specifies the filter date type as follows:

112	AssignedDate
126	BeginDateTime
139	CreateDate
507	DeliveredDate
148	DueDate
151	EndDateTime
331	NamedField

FieldName As ANSISTRING

(Optional)

UserID As ANSISTRING

(Optional)

Operation As ENUM

Specifies the operation as follows:

320	Equal
321	GT
322	GTE
318	LT
319	LTE

DayOfMonth As WORD

Month As WORD

Range from 0-11 (not 1-12).

Year As WORD

FilterSetDWord()

Filters items which have the user-defined field FieldName associated with it based on the FieldOperator and DWord Value specified.

Token ID

`AFTKN_FILTER_SET_DWORD or 838`

Syntax

```
VOID FilterSetDWord(ENUM FilterHandleType;
                    ANSISTRING FieldName;
                    ENUM FieldOperator;
                    DWORD FieldDWordValue;
                    [ANSISTRING UserID])
```

Parameters

FilterHandleType As ENUM

Unique filter identifier, returned by FilterCreate():

315 Default

FieldName As ANSISTRING

FieldOperator As ENUM

Specifies the field operator as follows:

147	DontCare
320	Equal
321	GT
322	GTE
318	LT
319	LTE
323	NotEqual

FieldDWordValue As DWORD

UserID As ANSISTRING

(Optional)

FilterSetItemType()

Filters items according to a specified item type.

Token ID

```
AFTKN_FILTER_SET_ITEMTYPE or 728
```

Syntax

```
VOID FilterSetItemType(ENUM FilterHandleType;
                       ENUM ItemType;
                       ENUM BitOn)
```

Parameters

FilterHandleType As ENUM

Unique filter identifier, returned by FilterCreate():

315 Default

ItemType As ENUM

Item type to filter, or not filter, depending on the BitOn setting. Use multiple commands to filter more than one item type. Item types include:

274	Appointment
276	Mail
278	Note
8	Phone
505	Profile
506	Search
277	Task

BitOn As ENUM

Specifies how to filter the item type, as follows:

147	DontCare
0	No
	Filters out the specified item type.
1	Yes
	Passes the specified item type through the filter, but no others.

Example

The following example allows Appointment! and Task! items to pass through a Mailbox filter:

```
hFilter = FilterCreate(Inbox!)
FilterSetItemType(Handle: hFilter; ItemType: Appointment!; BitOn: Yes!)
FilterSetItemType(Handle: hFilter; ItemType: Task!; BitOn: Yes!)
FilterApply(Handle: hFilter; ApplyTo: ItemList!)
FilterDelete(Handle: hFilter)
```

FilterSetPriority()

Filters items according to priority.

Token ID

```
AFTKN_FILTER_SET_PRIORITY or 729
```

Syntax

```
VOID FilterSetPriority(ENUM FilterHandleType;
                       ENUM Priority;
                       ENUM BitOn)
```

Parameters

FilterHandleType As ENUM

Unique filter identifier, returned by FilterCreate():

315 Default

Priority As ENUM

Priority to filter, or not to filter, depending on the BitOn setting. Use multiple commands to filter more than one priority. Priorities are as follows:

0	Low
1	Normal
2	High

BitOn As ENUM

Specifies how to filter the item type, as follows:

147	DontCare
0	No Filters out the specified item type.
1	Yes Passes the specified item type through the filter, but no others.

FilterSetSDWord()

Filters items which have the user-defined field FieldName associated with it based on the FieldOperator and SDWord Value specified.

Token ID

```
AFTKN_FILTER_SET_SDWORD or 827
```

Syntax

```
VOID FilterSetSDWord(ENUM FilterHandleType;
                     ANSISTRING FieldName;
                     ENUM FieldOperator;
                     DWORD FieldSDWordValue
                     [ANSISTRING UserID])
```

Parameters

FilterHandleType As ENUM

Unique filter identifier, returned by FilterCreate():

315 Default

FieldName As ANSISTRING

FieldOperator As ENUM

Specifies the field operator, as follows:

147	DontCare
320	Equal
321	GT
322	GTE
318	LT
319	LTE
323	NotEqual

FieldSDWordValue As DWORD

UserID As ANSISTRING

(Optional)

FilterSetSource()

Filters items according to their source. You must specify a source if ItemCreate created an empty filter.

Token ID

```
AFTKN_FILTER_SET_SOURCE or 730
```

Syntax

```
VOID FilterSetSource(ENUM FilterHandleType;
                     ENUM BoxType;
                     ENUM BitOn)
```

Parameters

FilterHandleType As ENUM

Unique filter identifier, returned by FilterCreate():

315 Default

BoxType As ENUM

Specifies the box type, as follows:

6	Inbox
7	Outbox
198	Personal

BitOn As ENUM

Specifies how to filter the item type, as follows:

147	DontCare
0	No
	Filters out the specified item type.
1	Yes
	Passes the specified item type through the filter, but no others.

FilterSetSWord()

FilterSetSWord() filters items which have the user-defined field FieldName associated with it based, on the FieldOperator and SWord Value specified.

Token ID

```
AFTKN_FILTER_SET_SWORD or 828
```

Syntax

```
VOID FilterSetSWord(ENUM FilterHandleType;
                    ANSISTRING FieldName;
                    ENUM FieldOperator;
                    WORD FieldSWordValue;
                    [ANSISTRING UserID])
```

Parameters

FilterHandleType As ENUM

Unique filter identifier, returned by FilterCreate():

315 Default

FieldName As ANSISTRING

FieldOperator As ENUM

Specifies the field operator, as follows:

147	DontCare
320	Equal
321	GT
322	GTE
318	LT
319	LTE
323	NotEqual

FieldSWordValue As WORD

UserID As ANSISTRING

(Optional)

FilterSetText()

Filters items according to the text in a text field.

Token ID

```
AFTKN_FILTER_SET_TEXT or 731
```

Syntax

```
VOID FilterSetText(ENUM FilterHandleType;
                   ENUM Field;
                   ANSISTRING FieldText;
                   ENUM Match;
                   [ANSISTRING FieldName];
                   [ANSISTRING UserID])
```

Parameters

FilterHandleType As ENUM

Unique filter identifier, returned by FilterCreate():

315 Default

Field As ENUM

Text field to filter. Use multiple commands to filter more than one field type. Text fields include:

625	Author!
4	Authority!
3	BC!
6	Caller!
	Specifies caller's Name
2	CC!
7	Company!
	Specifies caller's Company
626	Creator
1	From!
10	Message!
	If Message is selected, the BeginText! and FullText! filtering options cannot be used in the fourth parameter.
331	NamedField!
	Specifies the custom Field name in the Optional FieldName parameter.

8	Phone! Specifies caller's phone number.
5	Place!
9	Subject!
227	TaskCategory!
230	TaskPriority!
0	To!
267	ViewName!

FieldText As ANSISTRING

Text to match.

Match As ENUM

(Optional) SearchString in relation to the field contents. Position of the text in a character string. Specify as follows:

127	BeginText! Matches a string from the first charter to the length of the search string. If Message is selected for the second parameter, the BeginText! and FullText! filtering options cannot be used.
165	FullText! Requires an exact match and is case-sensitive. If Message is selected for the second parameter, the BeginText! and FullText! filtering options cannot be used.
225	SubText! Default. Matches any substrings.
659	NotContainText!

FieldName As ANSISTRING

(Optional)

UserID As ANSISTRING

(Optional)

Remarks

NamedFields is not available as ENUMS. Instead, use the NamedField (331) ENUM and specify the string in quotes as the FieldName.

For example, FilterSetText(315; 331; "Document"; 225; "NGW_HWZ_DOC_TYPE")
Document Type "NGW_HWZ_DOC_TYPE"Filename Extension
"NGW_HWZ_FILENAME"Version Description
"NGW_HWZ_VERSION_DESCRIPTION"Subclass
"NGW_HWZ_ITEM_SUBTYPE"Posted By
"NGW_HSZ_SHARER_FULL_NAME"Caller's Phone Number
"CALLER_PHONE_NUMBER"Caller's Company
"CALLER_COMPANY_TEXT"Library

"NGW_HWZ_LIB_DISPLAY_NAME"Opened By
"NGW_HWZ_VER_RET_BY_DISPLAY_NAME"Version Creator
"NGW_HWZ_VER_CREATOR_DISPLAY_NAME"

FilterSetVersionStatus()

Filters for document reference items which have a version status matching the one specified in VersionStatus.

Token ID

```
AFTKN_FILTER_SET_VERSTATUS or 873
```

Syntax

```
VOID FilterSetVersionStatus(ENUM FilterHandleType;
                            ENUM VersionStatus;
                            ENUM BitOn)
```

Parameters

FilterHandleType As ENUM

Unique filter identifier, returned by FilterCreate():

315 Default

VersionStatus As ENUM

Specifies the version status as follows:

279	Archived
640	Available
641	CheckedOut
642	InUse
643	RemoteInUse
644	MassInUse

BitOn As ENUM

Specifies how to filter the item type as follows:

147	DontCare
0	No
	Filters out the specified item type.
1	Yes
	Passes the specified item type through the filter, but no others.

FilterSetWord()

Filters items which have the user-defined field FieldName associated with it based, on the FieldOperator and Word Value specified.

Token ID

```
AFTKN_FILTER_SET_WORD or 839
```

Syntax

```
VOID FilterSetWord(ENUM FilterHandleType;
                   ANSISTRING FieldName;
                   ENUM FieldOperator;
                   WORD FieldWordValue;
                   [ANSISTRING UserID])
```

Parameters

FilterHandleType As ENUM

Unique filter identifier, returned by FilterCreate():

315 Default

FieldName As ANSISTRING

FieldOperator As ENUM

Specifies the field operator as follows:

147	DontCare
320	Equal
321	GT
322	GTE
318	LT
319	LTE
323	NotEqual

FieldWordValue As WORD

UserID As ANSISTRING

(Optional)

Find()

Finds a character string in the From, Subject, or Message edit boxes.

Token ID

```
AFTKN_SEARCH or 627
```

Syntax

```
VOID Find([ANSISTRING FromSearchString];
          [ANSISTRING SubjectSearchString];
          [ANSISTRING MessageSearchString];
          [ENUM SearchDir];
          [ENUM FromMatch];
          [ENUM SubjectMatch];
          [ENUM MessageMatch])
```

Parameters

FromSearchString As ANSISTRING

(Optional) From edit box character string.

SubjectSearchString As ANSISTRING

(Optional) Subject edit box character string.

MessageSearchString As ANSISTRING

(Optional) Message edit box character string.

SearchDir As ENUM

(Optional) Search forward or backward from the insertion point:

0 Forward (Default)
1 Backward

FromMatch As ENUM

(Optional) From edit box match criteria, as follows:

Value	Position	Description
127	BeginText	Matches a string from the first character to the length of the search string.
165	FullText	Matches an exact string and is case-sensitive.
225	SubText	Default. Matches any substring.

SubjectMatch As ENUM

(Optional) Match criteria for the Subject edit box, as follows:

Value	Position	Description
127	BeginText	Matches a string from the first character to the length of the search string.
165	FullText	Matches an exact string and is case-sensitive.
225	SubText	Default. Matches any substring.

MessageMatch As ENUM

(Optional) Match criteria for the Message edit box, as follows:

Value	Position	Description
127	BeginText	Matches a string from the first character to the length of the search string.
165	FullText	Matches an exact string and is case-sensitive.
225	SubText	Default. Matches any substring.

Remarks

A NotFound condition occurs if you use FromSearchString, SubjectSearchString, and MessageSearchString, and one or more does not find a match.

When you search only one edit box, you must include the parameter name. For example, the following line searches forward from the insertion point for "John" in a From edit box.

```
Find(FromSearchString: "John"; SearchDir: Forward!; FromMatch: SubText!)
```

See Also

“FindNext()” on page 354
“FindPrevious()” on page 355

FindContacts()

Displays a dialog enabling the user to search for contacts.

Token ID

```
DTKN_FIND_CONTACTS or 152
```

Syntax

```
VOID FindContacts([ANSISTRING] SearchText)
```

Parameters

SearchText As ANSISTRING

(Optional) The text used for the initial search.

FindDlg()

Opens the Find dialog box to search the Message edit box of the current view, or displays an error message if the Message edit box does not have the input focus.

Token ID

```
DTKN_SEARCH or 67
```

Syntax

```
VOID FindDlg()
```

FindNext()

Finds the next occurrence of a character string.

Token ID

```
BFTKN_SEARCH_NEXT or 316
```

Syntax

```
VOID FindNext()
```

FindPrevious()

Finds the previous occurrence of a character string.

Token ID

```
BFTKN_SEARCH_PREV or 317
```

Syntax

```
VOID FindPrevious()
```

FindVacationRule()

Returns the name of the vacation rule.

Token ID

```
BFTKN_FIND_VACATION_RULE or 1250
```

Syntax

```
ANSISTRING FindVacationRule()
```

Return Values

ANSISTRING. Returns the name of the vacation rule that GroupWise creates and/or modifies when invoking the Tools > Vacation Rule menu item.

FocusSet()

Gives the input focus to a specified Windows control.

Token ID

```
AFTKN_SETFOCUS or 617
```

Syntax

```
VOID FocusSet(ENUM Place;
              [ANSISTRING ControlName])
```

Parameters

Place As ENUM

Control to receive the input focus, as follows:

11	AcceptBtn
24	AddressBtn
12	AttachBtn
27	Attachments
4	Authority
3	BC
25	BusyBtn
29	Called
6	Caller
34	CameToSeeYou
13	CancelOrCloseBtn
2	CC
7	Company
273	ControlName
36	DayAppt
37	DayNote
38	DayTask
20	DeclineBtn
14	DelegateBtn
15	DeleteBtn
43	FolderList

349	FollowUpTo
16	ForwardBtn
1	From
17	InfoBtn
44	ItemList
504	LargeMonth
10	Message
18	OkBtn
8	Phone
5	Place
30	PleaseCall
19	PrintBtn
26	Priority
21	ReplyBtn
32	ReturnedYourCall
22	SaveBtn
28	Security
23	SendBtn
42	SmallMonth
9	Subject
0	To
35	Urgent
33	WantsToSeeYou
39	WeekAppt
40	WeekNote
41	WeekTask
31	WillCall

ControlName As ANSISTRING

(Optional) Name of a user-defined control.

Remarks

Note: This is not an option for Start/End Date, Duration, or Custom button controls.

FolderAddToFavorites()

Adds the currently selected folder to the favorites folder list.

Token ID

```
BFTKN_ADD_TO_FAVORITES or 1280
```

Syntax

```
VOID FolderAddToFavorites()
```

FolderContract()

Contracts a specified folder (hides all subfolders).

Token ID

```
BFTKN_FOLDER_CONTRACT or 297
```

Syntax

```
VOID FolderContract([ANSISTRING FolderName])
```

Parameters

FolderName As ANSISTRING

(Optional) Include full folder path.

See Also

“FolderExpand()” on page 369
“FolderExpandAll()” on page 370

FolderContractToOneLevel()

Contracts a specific folder to one level (hides subfolders). There must be two levels of subfolders.

Token ID

```
BFTKN_FOLDER_CONTRACTEX or 298
```

Syntax

```
VOID FolderContractToOneLevel([ANSISTRING FolderName])
```

Parameter

FolderName As ANSISTRING

(Optional) Include full folder path.

See Also

“FolderExpand()” on page 369
“FolderExpandAll()” on page 370

FolderCreate()

Creates a new folder.

Token ID

```
AFTKN_CREATE_FOLDER or 562
```

Syntax

```
VOID FolderCreate(ANSISTRING FolderName;
                  ANSISTRING FolderDesc;
                  WORD FolderPosition)
```

Parameters

FolderName As ANSISTRING

Include full folder path.

FolderDesc As ANSISTRING

FolderPosition As WORD

See Also

“FolderExpand()” on page 369
“FolderExpandAll()” on page 370

FolderCreateDlg()

Displays the folder name dialog box to create a new folder. The command fails if a folder item is selected instead of a folder.

Token ID

```
DTKN_CREATE_FOLDER or 36
```

Syntax

```
VOID FolderCreateDlg([ANSISTRING FolderName])
```

Parameters

FolderName As ANSISTRING

(Optional) Include full folder path.

See Also

“FolderExpand()” on page 369
“FolderExpandAll()” on page 370

FolderDeepMove()

Deletes all links and moves selected items to one or more folders (max = 50).

Token ID

```
AFTKN_ALTMOVETO_FOLDER or 623
```

Syntax

```
VOID FolderDeepMove(ANSISTRING FolderName00;
                    [ANSISTRING FolderName01];
                    [ANSISTRING FolderName02];
                    [ANSISTRING FolderName03];
                    [ANSISTRING FolderName04];
                    [ANSISTRING FolderName05];
                    [ANSISTRING FolderName06];
                    [ANSISTRING FolderName07];
                    [ANSISTRING FolderName08];
                    [ANSISTRING FolderName09];
                    [ANSISTRING FolderName10];
                    [ANSISTRING FolderName11];
                    [ANSISTRING FolderName12];
                    [ANSISTRING FolderName13];
                    [ANSISTRING FolderName14];
                    [ANSISTRING FolderName15];
                    [ANSISTRING FolderName16];
                    [ANSISTRING FolderName17];
                    [ANSISTRING FolderName18];
                    [ANSISTRING FolderName19];
                    [ANSISTRING FolderName20];
                    [ANSISTRING FolderName21];
                    [ANSISTRING FolderName22];
                    [ANSISTRING FolderName23];
                    [ANSISTRING FolderName24];
                    [ANSISTRING FolderName25];
                    [ANSISTRING FolderName26];
                    [ANSISTRING FolderName27];
                    [ANSISTRING FolderName28];
                    [ANSISTRING FolderName29];
                    [ANSISTRING FolderName30];
                    [ANSISTRING FolderName31];
                    [ANSISTRING FolderName32];
                    [ANSISTRING FolderName33];
                    [ANSISTRING FolderName34];
                    [ANSISTRING FolderName35];
                    [ANSISTRING FolderName36];
```

```
[ANSISTRING FolderName37];
[ANSISTRING FolderName38];
[ANSISTRING FolderName39];
[ANSISTRING FolderName40];
[ANSISTRING FolderName41];
[ANSISTRING FolderName42];
[ANSISTRING FolderName43];
[ANSISTRING FolderName44];
[ANSISTRING FolderName45];
[ANSISTRING FolderName46];
[ANSISTRING FolderName47];
[ANSISTRING FolderName48];
[ANSISTRING FolderName49])
```

Parameters

FolderName00 As ANSISTRING

Include full folder path.

FolderName01As ANSISTRING through FolderName49 As ANSISTRING

(Optional) Include full folder path. Separate multiple folders with semicolons.

FolderDelete()

Deletes one or more folders (max = 50) and their contents, or only the contents.

Token ID

```
AFTKN_DELETE_FOLDER or 564
```

Syntax

```
VOID FolderDelete(ENUM DeleteType;
                  ANSISTRING FolderName00;
                  [ANSISTRING FolderName01];
                  [ANSISTRING FolderName02];
                  [ANSISTRING FolderName03];
                  [ANSISTRING FolderName04];
                  [ANSISTRING FolderName05];
                  [ANSISTRING FolderName06];
                  [ANSISTRING FolderName07];
                  [ANSISTRING FolderName08];
                  [ANSISTRING FolderName09];
                  [ANSISTRING FolderName10];
                  [ANSISTRING FolderName11];
                  [ANSISTRING FolderName12];
                  [ANSISTRING FolderName13];
                  [ANSISTRING FolderName14];
                  [ANSISTRING FolderName15];
                  [ANSISTRING FolderName16];
                  [ANSISTRING FolderName17];
                  [ANSISTRING FolderName18];
                  [ANSISTRING FolderName19];
                  [ANSISTRING FolderName20];
                  [ANSISTRING FolderName21];
                  [ANSISTRING FolderName22];
                  [ANSISTRING FolderName23];
                  [ANSISTRING FolderName24];
                  [ANSISTRING FolderName25];
                  [ANSISTRING FolderName26];
                  [ANSISTRING FolderName27];
                  [ANSISTRING FolderName28];
                  [ANSISTRING FolderName29];
                  [ANSISTRING FolderName30];
                  [ANSISTRING FolderName31];
                  [ANSISTRING FolderName32];
                  [ANSISTRING FolderName33];
                  [ANSISTRING FolderName34];
                  [ANSISTRING FolderName35];
                  [ANSISTRING FolderName36];
```

```
[ANSISTRING FolderName37];
[ANSISTRING FolderName38];
[ANSISTRING FolderName39];
[ANSISTRING FolderName40];
[ANSISTRING FolderName41];
[ANSISTRING FolderName42];
[ANSISTRING FolderName43];
[ANSISTRING FolderName44];
[ANSISTRING FolderName45];
[ANSISTRING FolderName46];
[ANSISTRING FolderName47];
[ANSISTRING FolderName48];
[ANSISTRING FolderName49])
```

Parameters

DeleteType As ENUM

1 Items and folders
0 Items only

FolderName00 As ANSISTRING

Include full folder path.

FolderName01 As ANSISTRING through FolderName49 As ANSISTRING

(Optional) Include full folder path. Separate multiple folders with semicolons.

See Also

“FolderRename()” on page 389
“FolderSelect()” on page 391

FolderDeleteDlg()

Displays the Delete Folder(s) dialog box. This command fails if a folder item is selected instead of a folder.

Token ID

```
DTKN_DELETE_FOLDERS or 50
```

Syntax

```
VOID FolderDeleteDlg()
```

FolderExpand()

Displays the subfolders of a contracted folder.

Token ID

```
BFTKN_FOLDER_EXPAND or 295
```

Syntax

```
VOID FolderExpand([ANSISTRING FolderName])
```

Parameters

FolderName As ANSISTRING

(Optional) Include full folder path.

See Also

“FolderContract()” on page 360
“FolderContractToOneLevel()” on page 361
“FolderExpandAll()” on page 370

FolderExpandAll()

Displays all subfolders of a contracted folder.

Token ID

```
BFTKN_FOLDER_EXPAND_ALL or 299
```

Syntax

```
VOID FolderExpandAll()
```

FolderExpandAllLevels()

Displays the subfolders of a selected folder.

Token ID

```
BFTKN_FOLDER_EXPANDEX or 296
```

Syntax

```
VOID FolderExpandAllLevels([ANSISTRING FolderName])
```

Parameters

FolderName As ANSISTRING

(Optional) Include full folder path.

FolderLinkTo()

Links an item to one or more folders (as many as 50). Linked items appear in more than one folder.

Token ID

```
AFTKN_LINKTO_FOLDER or 622
```

Syntax

```
VOID FolderLinkTo(ANSISTRING FolderName00;
                  [ANSISTRING FolderName01];
                  [ANSISTRING FolderName02];
                  [ANSISTRING FolderName03];
                  [ANSISTRING FolderName04];
                  [ANSISTRING FolderName05];
                  [ANSISTRING FolderName06];
                  [ANSISTRING FolderName07];
                  [ANSISTRING FolderName08];
                  [ANSISTRING FolderName09];
                  [ANSISTRING FolderName10];
                  [ANSISTRING FolderName11];
                  [ANSISTRING FolderName12];
                  [ANSISTRING FolderName13];
                  [ANSISTRING FolderName14];
                  [ANSISTRING FolderName15];
                  [ANSISTRING FolderName16];
                  [ANSISTRING FolderName17];
                  [ANSISTRING FolderName18];
                  [ANSISTRING FolderName19];
                  [ANSISTRING FolderName20];
                  [ANSISTRING FolderName21];
                  [ANSISTRING FolderName22];
                  [ANSISTRING FolderName23];
                  [ANSISTRING FolderName24];
                  [ANSISTRING FolderName25];
                  [ANSISTRING FolderName26];
                  [ANSISTRING FolderName27];
                  [ANSISTRING FolderName28];
                  [ANSISTRING FolderName29];
                  [ANSISTRING FolderName30];
                  [ANSISTRING FolderName31];
                  [ANSISTRING FolderName32];
                  [ANSISTRING FolderName33];
                  [ANSISTRING FolderName34];
                  [ANSISTRING FolderName35];
                  [ANSISTRING FolderName36];
```

```
[ANSISTRING FolderName37];
[ANSISTRING FolderName38];
[ANSISTRING FolderName39];
[ANSISTRING FolderName40];
[ANSISTRING FolderName41];
[ANSISTRING FolderName42];
[ANSISTRING FolderName43];
[ANSISTRING FolderName44];
[ANSISTRING FolderName45];
[ANSISTRING FolderName46];
[ANSISTRING FolderName47];
[ANSISTRING FolderName48];
[ANSISTRING FolderName49])
```

Parameters

FolderName00 As ANSISTRING

Include full folder path.

FolderName01 As ANSISTRING through FolderName49

As ANSISTRING are optional. Include full folder path. Separate multiple folders with semicolons.

FolderLinkToDlg()

Displays the Mailbox Selections, Sent Items Selections, or Move/Link Selections to Folder dialog box, depending on the folder list.

Token ID

```
DTKN_LINK_FOLDER or 78
```

Syntax

```
VOID FolderLinkToDlg()
```

See Also

“FolderLinkTo()” on page 372

FolderListAddFolder()

Adds one or more folders to a folder list.

Token ID

```
AFTKN_FLDRLIST_ADD or 702
```

Syntax

```
VOID FolderListAddFolder(DWORD FolderListHandle;
                         ANSISTRING FolderName)
```

Parameters

FolderListHandle As DWORD

Folder list handle, returned by FolderListCreate As ANSISTRING or FolderListCreateFromView As ANSISTRING.

FolderName As ANSISTRING

Include full folder path.

FolderListApplyToView()

Applies a folder list to an active view.

Token ID

```
AFTKN_FLDRLIST_APPLY or 703
```

Syntax

```
VOID FolderListApplyToView(DWORD FolderListHandle)
```

Parameters

FolderListHandle As DWORD

Folder list handle, returned by FolderListCreate or FolderListCreateFromView.

FolderListCreate()

Creates a folder list.

Token ID

```
AFTKN_FLDRLIST_CREATE or 704
```

Syntax

```
DWORD FolderListCreate(ENUM Contents;
                       [ANSISTRING UserID])
```

Parameters

Contents As ENUM

Create a folder list including all folders, no folders, or only the root folder. If you create an Empty! folder list, call FolderListAddFolder() before passing the folder list handle to ItemListCreate. Specify as follows:

7	All
150	Empty
216	Root

UserID As ANSISTRINGA

(Optional) User ID (root folder).

Return Values

DWORD. Folder list handle if successful, 0 if not.

See Also

“FolderListCreateFromView()” on page 378
“ItemListCreate()” on page 493

Remarks

IMPORTANT: Delete a folder list handle when it is no longer needed, or system resources will be lost. (See FolderListDelete.)

FolderListCreateFromView()

Creates a folder list from one or more selected folders (active view).

Token ID

```
AFTKN_FLDRLIST_CREATE_FROM_VIEW or 705
```

Syntax

```
DWORD FolderListCreateFromView()
```

Return Values

DWORD. Folder list handle if successful, 0 if not.

Remarks

IMPORTANT: Delete a folder list handle when it is no longer needed, or system resources will be lost. (See FolderListDelete.)

See Also

“FolderListCreate()” on page 377

FolderListDelete()

Deletes a folder list handle from memory. The folder list handle is returned by FolderListCreate or FolderListCreateFromView.

Token ID

```
AFTKN_FLDRLIST_DELETE or 706
```

Syntax

```
VOID FolderListDelete(DWORD FolderListHandle)
```

Parameters

FolderListHandle As DWORD

Remarks

IMPORTANT: Delete a file folder list handle after it is used to create an item list, or when it is no longer needed, or system resources will be lost.

FolderListGetCount()

Counts the number of folders in a folder list created by FolderListCreate() or FolderListCreateFromView().

Token ID

```
AFTKN_FLDRLIST_GET_Count or 707
```

Syntax

```
WORD FolderListGetCount (DWORD FolderListHandle)
```

Parameters

FolderListHandle As DWORD

Folder list handle, returned by FolderListCreate or FolderListCreateFromView.

Return Values

WORD. Returns the number of folders in the folder list.

FolderListGetName()

Returns a folder name from a folder list.

Token ID

```
AFTKN_FLDRLIST_GET_NAME or 708
```

Syntax

```
ANSISTRING FolderListGetName(DWORD FolderListHandle;
                             WORD Index;
                             ENUM FullyQualified)
```

Parameters

FolderListHandle As DWORD

Folder list handle, returned by FolderListCreate or FolderListCreateFromView.

Index As WORD

Index number (first folder equals 0).

FullyQualified As ENUM

Include full folder path:

0 No
1 Yes

Return Values

ANSISTRING. Folder name.

FolderListOrderDlg()

Displays a dialog that allows the user to configure the display order of the full, simple, and favorites list.

Token ID

```
DTKN_FOLDER_TREE_ORDER or 154
```

Syntax

```
VOID FolderListOrderDlg()
```

Remarks

Note that this token is only enabled when more than one type of folder list (full, simple, favorites) is showing.

FolderListSetFolder()

Replaces the contents of a folder list with one or more specified folders (creates a new list).

Token ID

```
AFTKN_FLDRLIST_SET or 709
```

Syntax

```
VOID FolderListSetFolder(DWORD FolderListHandle;
                         ANSISTRING FolderName)
```

Parameters

FolderListHandle As DWORD

Handle of the folder list that has contents to replace. The handle is returned by FolderListCreate or FolderListCreateFromView.

FolderName As ANSISTRING

Folder name to add.

See Also

“EnvUserID()” on page 298

FolderMoveTo()

Moves an item to one or more folders and removes it from the original folder.

Token ID

```
AFTKN_MOVETO_FOLDER or 621
```

Syntax

```
VOID FolderMoveTo(ANSISTRING FolderName00;
                      [ANSISTRING FolderName01];
                      [ANSISTRING FolderName02];
                      [ANSISTRING FolderName03];
                      [ANSISTRING FolderName04];
                      [ANSISTRING FolderName05];
                      [ANSISTRING FolderName06];
                      [ANSISTRING FolderName07];
                      [ANSISTRING FolderName08];
                      [ANSISTRING FolderName09];
                      [ANSISTRING FolderName10];
                      [ANSISTRING FolderName11];
                      [ANSISTRING FolderName12];
                      [ANSISTRING FolderName13];
                      [ANSISTRING FolderName14];
                      [ANSISTRING FolderName15];
                      [ANSISTRING FolderName16];
                      [ANSISTRING FolderName17];
                      [ANSISTRING FolderName18];
                      [ANSISTRING FolderName19];
                      [ANSISTRING FolderName20];
                      [ANSISTRING FolderName21];
                      [ANSISTRING FolderName22];
                      [ANSISTRING FolderName23];
                      [ANSISTRING FolderName24];
                      [ANSISTRING FolderName25];
                      [ANSISTRING FolderName26];
                      [ANSISTRING FolderName27];
                      [ANSISTRING FolderName28];
                      [ANSISTRING FolderName29];
                      [ANSISTRING FolderName30];
                      [ANSISTRING FolderName31];
                      [ANSISTRING FolderName32];
                      [ANSISTRING FolderName33];
                      [ANSISTRING FolderName34];
                      [ANSISTRING FolderName35];
                      [ANSISTRING FolderName36];
```

```
[ANSISTRING FolderName37];
[ANSISTRING FolderName38];
[ANSISTRING FolderName39];
[ANSISTRING FolderName40];
[ANSISTRING FolderName41];
[ANSISTRING FolderName42];
[ANSISTRING FolderName43];
[ANSISTRING FolderName44];
[ANSISTRING FolderName45];
[ANSISTRING FolderName46];
[ANSISTRING FolderName47];
[ANSISTRING FolderName48];
[ANSISTRING FolderName49])
```

Parameters

FolderName00 As ANSISTRING

Include full folder path

FolderName01 As ANSISTRING through FolderName49 As ANSISTRING

(Optional) Include full folder path. Separate multiple folders with semicolons.

See Also

“FolderDeepMove()” on page 364
“FolderLinkTo()” on page 372
“FolderSelect()” on page 391

FolderParent()

Selects the parent of the current folder.

Token ID

```
BFTKN_PARENT or 356
```

Syntax

```
VOID FolderParent()
```

FolderProfileReferenceDlg()

Displays dialog allowing the user to add a document reference to the selected folder.

Token ID

```
DTKN_FOLDER_PROFILEREF or 87
```

Syntax

```
VOID FolderProfileReferenceDlg()
```

FolderRemoveFromFavorites()

Removes the currently selected folder from the favorites folder list.

Token ID

```
BFTKN_REMOVE_FROM_FAVORITES or 1284
```

Syntax

```
VOID FolderRemoveFromFavorites()
```

FolderRename()

Renames a folder.

Token ID

```
AFTKN_RENAME_FOLDER or 565
```

Syntax

```
VOID FolderRename(ANSISTRING FullFolderName;
                  ANSISTRING NewName)
```

Parameters

FullFolderName As ANSISTRING

Folder to rename. Include full path.

NewName As ANSISTRING

New folder name only. Do not include full path.

FolderRenameDlg()

Displays the Rename Folder dialog box. The command fails if a folder item is selected instead of a folder.

Token ID

```
DTKN_RENAME_FOLDER or 35
```

Syntax

```
VOID FolderRenameDlg([ANSISTRING FolderName])
```

Parameters

FolderName As ANSISTRING

(Optional) If not specified, folder must be selected.

FolderSelect()

Selects up to 50 folders. Each new selection deselects the previous section.

Token ID

```
BFTKN_SELECT_FOLDERS or 363
```

Syntax

```
VOID FolderSelect(ANSISTRING FolderName00
                  [ANSISTRING FolderName01];
                  [ANSISTRING FolderName02];
                  [ANSISTRING FolderName03];
                  [ANSISTRING FolderName04];
                  [ANSISTRING FolderName05];
                  [ANSISTRING FolderName06];
                  [ANSISTRING FolderName07];
                  [ANSISTRING FolderName08];
                  [ANSISTRING FolderName09];
                  [ANSISTRING FolderName10];
                  [ANSISTRING FolderName11];
                  [ANSISTRING FolderName12];
                  [ANSISTRING FolderName13];
                  [ANSISTRING FolderName14];
                  [ANSISTRING FolderName15];
                  [ANSISTRING FolderName16];
                  [ANSISTRING FolderName17];
                  [ANSISTRING FolderName18];
                  [ANSISTRING FolderName19];
                  [ANSISTRING FolderName20];
                  [ANSISTRING FolderName21];
                  [ANSISTRING FolderName22];
                  [ANSISTRING FolderName23];
                  [ANSISTRING FolderName24];
                  [ANSISTRING FolderName25];
                  [ANSISTRING FolderName26];
                  [ANSISTRING FolderName27];
                  [ANSISTRING FolderName28];
                  [ANSISTRING FolderName29];
                  [ANSISTRING FolderName30];
                  [ANSISTRING FolderName31];
                  [ANSISTRING FolderName32];
                  [ANSISTRING FolderName33];
                  [ANSISTRING FolderName34];
                  [ANSISTRING FolderName35];
                  [ANSISTRING FolderName36];
```

```
[ANSISTRING FolderName37];
[ANSISTRING FolderName38];
[ANSISTRING FolderName39];
[ANSISTRING FolderName40];
[ANSISTRING FolderName41];
[ANSISTRING FolderName42];
[ANSISTRING FolderName43];
[ANSISTRING FolderName44];
[ANSISTRING FolderName45];
[ANSISTRING FolderName46];
[ANSISTRING FolderName47];
[ANSISTRING FolderName48];
[ANSISTRING FolderName49])
```

Parameters

FolderName00 As ANSISTRING

Include full folder path.

FolderName01 As ANSISTRING through FolderName49 As ANSISTRING

(Optional) Include full folder path. Separate multiple folders with semicolons.

Remarks

IMPORTANT: Selecting multiple folders had application in the GW 4.x, 5.x, and 6.x 16-bit clients. Selecting multiple folders at once in the GW 32-bit clients is not supported.

See Also

FolderMoveTo() (page 384)

FolderSelectDlg()

Displays the Folders dialog box.

Token ID

```
DTKN_SELECT_FOLDER or 46
```

Syntax

```
VOID FolderSelectDlg()
```

See Also

“FolderSelect()” on page 391

FollowInternetLink()

Launches browser to go to URL or create a new message addressed to the MAILTO address.

Token ID

```
BFTKN_FOLLOW_INTERNET_LINK or 474
```

Syntax

```
VOID FollowInternetLink([ANSISTRING Linkname];
                        [ENUM LinkType])
```

Parameters

Linkname As ANSISTRING

(Optional) Either the URL (http://www.aol.com/) or email address (mailto:). If omitted, assumes that a URL is under the current cursor position.

LinkType As ENUM

(Optional) Must be specified if Linkname is also specified.

1 BROWSER_LINK! For Internet URL.

2 MAIL_LINK! For MailTo: link.

FontBold()

Turns Bold on or off.

Token ID

```
BFTKN_BOLD or 248
```

Syntax

```
VOID FontBold([ENUM State])
```

Parameters

State As ENUM

(Optional) If not specified, acts as a toggle:

0 Off!
1 On!

See Also

“FontItalic()” on page 397
“FontNormal()” on page 398
“FontUnderline()” on page 402

FontDlg()

Displays the Font dialog box.

Token ID

```
DTKN_CHOOSE_FONT or 69
```

Syntax

```
VOID FontDlg()
```

See Also

“FontSet()” on page 399

FontItalic()

Turns Italic on or off.

Token ID

```
BFTKN_ITALIC or 250
```

Syntax

```
VOID FontItalic([ENUM State])
```

Parameters

State As ENUM

(Optional) If not specified, acts as a toggle:

0 Off!
1 On!

See Also

“FontBold()” on page 395
“FontNormal()” on page 398
“FontUnderline()” on page 402

FontNormal()

Turns off all current font attributes (Bold, Italic, or Underline).

Token ID

```
BFTKN_NORMAL or 251
```

Syntax

```
VOID FontNormal()
```

See Also

“FontBold()” on page 395
“FontItalic()” on page 397
“FontUnderline()” on page 402

FontSet()

Specifies a font and font attributes. To see the height a weight values of a specified font, record your actions as you specify a font and font attributes.

Token ID

`AFTKN_SET_FONT or 583`

Syntax

```
VOID FontSet(ANSISTRING FaceName;
             [DWORD Height];
             [DWORD Weight];
             [ENUM Underline];
             [ENUM Italic];
             [WORD CharSet];
             [ENUM Strikeout];
             [DWORD Color])
```

Parameters

FaceName As ANSISTRING

Name of font.

Height As DWORD

(Optional) Font height in logical height units. You can convert from pixels to logical height units using the following formula: `lfHeight = -MulDiv(PointSize, GetDeviceCaps(hDC, LOGPIXELSY), 72);`. Examples: Cell height: 0-32767. Character height including ascender and descender: 32768-65535.

Weight As DWORD

(Optional) Font weight. Some fonts have only three weights: Normal, Regular, and Bold.

100	THIN
200	EXTRATHIN
300	ULTRALIGHT
400	NORMAL
400	REGULAR
500	MEDIUM
600	SEMIBOLD
600	DEMIBOLD
700	BOLD
800	EXTRABOLD
800	ULTRABOLD
900	BLACK
900	HEAVY

Underline As ENUM

(Optional) Enumerated values:

0 No
1 Yes

Italic As ENUM

(Optional) Enumerated values:

0 No
1 Yes

CharSet As WORD

(Optional) Asian versions only.

Strikeout As ENUM

(Optional) Enumerated values:

0 No
1 Yes

Color As DWORD

(Optional) Use RGB values.

See Also

“FontBold()” on page 395
“FontItalic()” on page 397
“FontNormal()” on page 398

“FontUnderline()” on page 402

FontUnderline()

Turns Underline on or off.

Token ID

```
BFTKN_UNDERLINE or 249
```

Syntax

```
VOID FontUnderline([ENUM State])
```

Parameters

State As ENUM

(Optional) If not specified, acts as a toggle:

0 Off!
1 On!

See Also

“FontBold()” on page 395
“FontItalic()” on page 397
“FontNormal()” on page 398

FrameContentIsVisible()

Determines whether the content of the specified frame has been turned off by using ShowFrameContents() (page 834).

Token ID

```
BFTKN_FRAME_CONTENT_IS_VISIBLE or 1073
```

Syntax

```
BOOLEAN FrameContentIsVisible (ENUM WhichFrame)
```

Parameters

WhichFrame As ENUM

Enumerated values:

1 FolderList
2 MessageList
3 Calendar
4 QuickViewer

Return Values

BOOLEAN

FrameIsVisible()

Determines whether the specified frame shows in the main application window.

Token ID

```
BFTKN_FRAME_IS_VISIBLE or 1072
```

Syntax

```
BOOLEAN FrameIsVisible (ENUM WhichFrame)
```

Parameters

WhichFrame As ENUM

Enumerated values:

1 FolderList
2 MessageList
3 Calendar
4 QuickViewer

Return Values

BOOLEAN

FullFolderList()

Displays the full folder list.

Token ID

```
BFTKN_FOLDER_TREE_FULL or 1228
```

Syntax

```
VOID FullFolderList()
```

G-H

This section contains information about the following tokens:

GetAddressBookData()

Returns Address Book information (backward compatibility only). Use Address Book commands to return Address Book information.

Token ID

```
AFTKN_GET_ABDATA or 606
```

Syntax

```
GetAddressBookData([VARIABLE MacroVariable];
                   ENUM Section;
                   ENUM Query;
                   [WORD FieldNum];
                   [ANSISTRING UserIDOrGroupName];
                   [ENUM Disambiguate])
```

Parameters

MacroVariable As VARIABLE

(Optional) Output variable assigned the return value of the Query parameter.

Section As ENUM

Address book section to query, as follows:

159	ExternalAddess! For other mail systems.
3	PersonalGroups!
2	PublicGroups!
1	Resources!
0	Users!

Query As ENUM

Query type, as follows:

Value	Query Type	Description
2	Entry	Returns field contents.
1	Field	Returns the field name.
0	FieldCount	Returns the number of fields in a specified section.

FieldNum As WORD

(Optional) Field number to query.

UserIDOrGroupName As ANSISTRING

(Optional) User ID or group name to query.

Disambiguate As ENUM

(Optional) Choose the correct user when two or more users have the same ID.

Value	Choice	Description
0	No	Redirects macro execution to an ONERROR label. The macro fails if an ONERROR label is not specified.
1	Yes	Displays the list of users and prompts you to choose one.

Return Values

Any.

See Also

“GetOfficeData()” on page 410

GetAppearance()

Returns appearance information.

Token ID

```
AFTKN_GET_APPEARANCE or 926
```

Syntax

```
GetAppearance(ENUM FromDataStore;
              ENUM Request)
```

Parameters

FromDataStore As ENUM

Where to return the information from, as follows:

true	Persistent data store
false	Current cache

Request As ENUM

The appearance attributes requested, as follows:

1	ShowMainMenu!
2	ShowNavbar!
3	ShowMainToolbar!
4	ShowFolderList!
5	ShowQV!
6	SimpleFolderListView!
7	LongFolderList!
8	QVPositionRight!

Return Values

BOOLEAN.

GetOfficeData()

Returns GroupWise, Mailbox, Sent Items, or Trash information (backward compatibility only). Use environment (ENV) command to return system information.

Token ID

```
AFTKN_GET_MAILDATA or 592
```

Syntax

```
GetOfficeData([VARIABLE MacroVariable];
              ENUM SystemVariable;
              [WORD IndexNumber])
```

Parameters

MacroVariable As VARIABLE

(Optional) Requested information is returned in this variable.

SystemVariable As ENUM

Returns a value for various types of GroupWise data in a variable, as follows:

0	ClipboardText
30	InterimVersion
24	IsItemAccepted
23	IsItemCompleted
26	IsItemPrivate
25	IsItemRouted
10	IsListItemAccepted
9	IsListItemCompleted
12	IsListItemDelegated
13	IsListItemPrivate
8	IsListItemRead
11	IsListItemRouted
22	IsViewPersonal
20	ItemAttachmentCount
29	ItemAttachmentName
28	ItemCreateDate
27	ItemPriority
6	ListCount

15	ListItemCreateDate
18	ListItemFrom
16	ListItemHasAttachments
14	ListItemPriority
19	ListItemSubject
7	ListItemType
3	MacroPath
1	MajorVersion
2	MinorVersion
17	PostOfficePath
4	UserId
5	UserName
21	ViewType

IndexNumber As WORD

(Optional) This method is not available in GroupWise 5.x and later Remote.

Return Values

ANY.

See Also

"GetAddressBookData()" on page 407

GetSetting()

Retrieves the specified user setting.

Token ID

```
BFTKN_GET_SETTING or 1071
```

Syntax

```
ANY GetSetting(ENUM WhichSetting)
```

Parameters

WhichSetting

Enumerated Values:

1	WorkScheduleStart - returns DWORD. The start of the work day in minutes since midnight.
2	WorkScheduleEnd - returns DWORD. The end of the work day in minutes since midnight.
3	WorkDays - returns DWORD. A bit array describing the days of the week that a user is scheduled to be working. The least significant bit represents Sunday and the most significant bit represents Saturday.
4	WorkDaySunday - returns BOOL. TRUE if Sunday is a workday.
5	WorkDayMonday - returns BOOL. TRUE if Monday is a workday.
6	WorkDayTuesday - returns BOOL. TRUE if Tuesday is a workday.
7	WorkDayWednesday - returns BOOL. TRUE if Wednesday is a workday.
8	WorkDayThursday - returns BOOL. TRUE if Thursday is a workday.
9	WorkDayFriday - returns BOOL. TRUE if Friday is a workday.
10	WorkDaySaturday - returns BOOL. TRUE is Saturday is a workday.

Return Values

ANY

HelpContents()

Displays the GroupWise Help Contents screen.

Token ID

```
BFTKN_HELP_CONTENTS or 168
```

Syntax

```
VOID HelpContents()
```

HelpCoolSolutions()

Launches the user's Web browser to the GroupWise Cool Solutions Web site.

Token ID

```
BFTKN_HELP_COOL_SOLUTIONS or 490
```

Syntax

```
VOID HelpCoolSolutions()
```

HelpNovellHomepage()

Links to the Novell home page on the World Wide Web.

Token ID

```
BFTKN_HELP_INTERNET or 475
```

Syntax

```
VOID HelpNovellHomepage()
```

HelpUsersGuide()

Invokes the User's Guide Help.

Token ID

```
BFTKN_HELP_USERS_GUIDE 1148
```

Syntax

```
void HelpUsersGuide()
```

HelpWhatsNew()

Invokes the What's New Help.

Token ID

```
BFTKN_HELP_WHATS_NEW 170
```

Syntax

```
VOID HelpWhatsNew()
```

I-K

This section contains information about the following tokens:

ImportContact()

Adds the specified file name to the default library specified in the GroupWise documents options.

Token ID

`BFTKN_IMPORT_CONTACT or 1248`

Syntax

```
VOID ImportContact()
```

ImportDocument()

Adds the specified file name to the default library specified in the GroupWise documents options.

Token ID

```
AFTKN_DM_IMPORT_DOC or 861
```

Syntax

```
ANSISTRING ImportDocument([ANSISTRING Filename])
```

Parameters

Filename As ANSISTRING

(Optional)

Return Values

ANSISTRING.

InfoUpdate()

Updates an item's Information view. The Item Properties view window must be displayed for this token to function.

Token ID

```
BFTKN_INFO_UPDATE or 318
```

Syntax

```
VOID InfoUpdate()
```

See Also

"Refresh()" on page 709

InsertTab()

Insert a tab.

Token ID

```
BFTKN_CTRL_TAB or 260
```

Syntax

```
VOID InsertTab()
```

Remarks

The message body field (in the Message View and the Object API) has a limit of 32,000 characters. The Message View will stop accepting data after 32,000 characters.

InvokeSpellerForWindow()

Runs the GroupWise spell checker for the edit window that is associated with the specified window handle (HWND).

Token ID

```
AFTKN_INVOKE_SPELLER_FOR_WINDOW or 916
```

Syntax

```
VOID InvokeSpellerForWindow(DWORD WindowHandle;
                            [BOOLEAN Destroy])
```

Parameters

WindowHandle As DWORD

The HWND of the window for which you want to run the spell checker. This window should be derived from an edit or richedit window class because it must provide meaningful responses to the following windows messages:

1) WM_GETTEXTLENGTH
2) WM_GETTEXT
3) EM_GETSEL
4) EM_SETSEL
5) EM_SCROLLCARET
6) EM_LINEFROMCHAR
7) EM_GETFIRSTVISIBLELINE
8) EM_LINESCROLL
9) EM_REPLACESEL

Destroy As BOOLEAN

(Optional) After you are done with the spell checker, call this token a second time with this flag set to TRUE to free up any resources that are being internally used by GroupWise.

ItemAccept()

Has been modified from the old ItemAccept token to take the MessageID and AllInstances parameters that allow a one-step completion of the action (with no input required from the user). If neither of these new parameters is specified, the token is internally rerouted to the ItemAcceptOpenItem() (page 427) token.

Token ID

```
AFTKN_ITEM_ACCEPT or 917
```

Syntax

```
VOID ItemAccept([ANSISTRING Comment];
                [long AcceptLevel;
                [ANSISTRING MessageID];
                [ENUM AllInstances])
```

Parameters

Comment As ANSISTRING

(Optional) Response to an accepted item.

AcceptLevel As LONG

(Optional) Even though this variable is a LONG, use the following values as you would an enumeration:

160 FREE
190 TENTATIVE
210 BUSY
251 OUT

MessageID As ANSISTRING

(Optional) Unique item identifier.

AllInstances As ENUM

(Optional)

Yes Accept all instances of this item
No Only accept THIS instance of this item (default)

ItemAcceptOpenItem()

Is the new text mnemonic that was given to the old ItemAccept() (page 426) token. Marks the currently open item as "Accepted."

Token ID

```
BFTKN_ACCEPT or 194
```

Syntax

```
VOID ItemAcceptOpenItem([ANSISTRING Comment];
                        [LONG AcceptLevel])
```

Parameters

Comment As ANSISTRING

(Optional) Response to an accepted item.

AcceptLevel As LONG

(Optional) Use these variables as you would an enumeration (even though this variable is a LONG):

160 FREE
190 TENTATIVE
210 BUSY
251 OUT

ItemAcceptWithCommentDlg()

Displays the Accept With Comment dialog box.

Token ID

`DTKN_ACCEPT_REPLY or 56`

Syntax

```
VOID ItemAcceptWithCommentDlg()
```

See Also

“ItemAccept()” on page 426

ItemAddMimeXField()

Adds a MIME X-field onto an item. This allows third-parties to store data onto any item in a key/value pair.

Token ID

```
AFTKN_ITEM_ADD_MIME_XFIELD or 930
```

Syntax

```
VOID ItemAddMimeXField(
     ANSISTRING MessageID;
     ANSISTRING XFieldName;
     ANSISTRING XFieldValue)
```

Parameters

MessageID As ANSISTRING

Unique item identifier, returned by ItemMessageIDFromView().

XFieldName As ANSISTRING

The name of the X-FIELD to set. For instance, “X-MYADDIN-SOMEVALUE”.

XFieldValue As ANSISTRING

The value of the X-FIELD to set. This can be any string value that is meaningful to your application.

ItemAnnotationGetCount()

Returns the number of sound annotations on an item view.

Token ID

```
AFTKN_ITEM_ANNOTATION_COUNT or 664
```

Syntax

```
WORD ItemAnnotationGetCount(ANSISTRING MessageID)
```

Parameters

MessageID As ANSISTRING

Unique Item identifier, returned by ItemMessageIDFromView.

Return Values

WORD. Number of sound annotations.

Remarks

IMPORTANT: This only works in the 32-bit client if the message with Annotation was created and sent by the 16-bit client.

ItemAnnotationSaveAs()

Saves a sound annotation to a file.

Token ID

```
AFTKN_ITEM_ANNOTATION_SAVE_AS or 663
```

Syntax

```
VOID ItemAnnotationSaveAs(ANSISTRING MessageID;
                          WORD Index;
                          ANSISTRING Filename)
```

Parameters

MessageID As ANSISTRING

Unique Item identifier, returned by ItemMessageIDFromView().

Index As WORD

Index number (zero-based) of the sound annotation.

Filename As ANSISTRING

Remarks

IMPORTANT: This only works in the 32-bit client if the message with Annotation was created and sent by the 16-bit client.

ItemArchive()

Saves an item to the archive database.

Token ID

```
AFTKN_ITEM_ARCHIVE or 785
```

Syntax

```
VOID ItemArchive([ANSISTRING MessageID])
```

Parameters

MessageID As ANSISTRING

(Optional) Unique item identifier, returned by ItemMessageIDFromView(). If not specified, archives the selected item.

Remarks

IMPORTANT: This token behaves like ItemArchiveOpenItem if no MessageID parameter is used. It also unarchives an item if in "Open Archive" mode. A selected item and using X00 as the MessageID work in both client states of regular and Open Archive mode. An item's MessageID changes after being archived and changes again when unarchived. The original MessageID is not used again when unarchived.

ItemArchiveOpenItem()

Saves one or more selected items to the archive database. This token also unarchives an item if in the archive mode.

Token ID

```
BFTKN_ARCHIVE or 293
```

Syntax

```
VOID ItemArchiveOpenItem()
```

See Also

“AttachmentAdd()” on page 103

ItemAttachmentAdd()

Adds an attachment to an item. Using X00 will work for the messageID if the item view is showing.

Token ID

```
AFTKN_ITEM_ATTACHMENT_ADD or 665
```

Syntax

```
VOID ItemAttachmentAdd(ANSISTRING MessageID;
                       ENUM AttachmentClass;
                       ANSISTRING AttachmentName;
                       [ANSISTRING AttachmentDisplayname])
```

Parameters

MessageID As ANSISTRING

Unique item identifier, returned by ItemMessageIDFromView().

AttachmentClass As ENUM

Type of attachment:

114 Attach Class File
115 Attach Class Message

AttachmentName As ANSISTRING

Name of an attachment.

AttachmentDisplayName As ANSISTRING

(Optional) Name displayed in the Attach File box.

See Also

"AttachmentAdd()" on page 103

ItemAttachmentDelete()

Deletes an attachment. Using X00 will work for the messageID if the item view is showing.

Token ID

```
AFTKN_ITEM_ATTACHMENT_DELETE or 666
```

Syntax

```
VOID ItemAttachmentDelete(ANSISTRING MessageID;
                          WORD Index)
```

Parameters

MessageID As ANSISTRING

Unique item identifier, returned by ItemMessageIDFromView().

Index As WORD

Attachment index number (zero-based).

See Also

"AttachmentDelete()" on page 105

ItemAttachmentGetClass()

Returns an attachment class. Using X00 will work for the messageID if the item view is showing.

Token ID

```
AFTKN_ITEM_ATTACHMENT_CLASS or 670
```

Syntax

```
WORD ItemAttachmentGetClass(ANSISTRING MessageID;
                            WORD Index)
```

Parameters

MessageID As ANSISTRING

Unique item identifier, returned by ItemMessageIDFromView().

Index As WORD

Return Values

WORD. Attachment Class:
1 File
2 Encapsulated item
3 Embedded or linked object (OLE)

ItemAttachmentGetCount()

Returns the number of attachments. Using X00 will work for the messageID if the item view is showing.

Token ID

```
AFTKN_ITEM_ATTACHMENT_COUNT or 667
```

Syntax

```
WORD ItemAttachmentGetCount(ANSISTRING MessageID)
```

Parameters

MessageID As ANSISTRING

Unique item identifier, returned by ItemMessageIDFromView().

Return Values

WORD.

ItemAttachmentGetCurrentIndex()

Returns the current attachment index. The item view must be open with the attachment selected.

Token ID

```
AFTKN_ITEM_ATTACHMENT_CURRENT or 758
```

Syntax

```
WORD ItemAttachmentGetCurrentIndex()
```

Return Values

WORD. Current attachment index (zero-based).

ItemAttachmentGetDisplayName()

Returns the display name of an attachment.

Token ID

```
AFTKN_ITEM_ATTACHMENT_DISP_NAME or 668
```

Syntax

```
ANSISTRING ItemAttachmentGetDisplayName(ANSISTRING MessageID;
                                        WORD Index)
```

Parameters

MessageID As ANSISTRING

Unique item identifier, returned by ItemMessageIDFromView().

Index As WORD

Attachment index number (zero-based).

Return Values

ANSISTRING. Display name, which may be different from the attachment file name.

ItemAttachmentGetName()

Returns the name of an attachment.

Token ID

```
AFTKN_ITEM_ATTACHMENT_NAME or 669
```

Syntax

```
ANSISTRING ItemAttachmentGetName(ANSISTRING MessageID;
                                 WORD Index)
```

Parameters

MessageID As ANSISTRING

Unique item identifier, returned by ItemMessageIDFromView().

Index As WORD

Attachment index number (zero-based).

Return Values

ANSISTRING. Attachment name.

Remarks

IMPORTANT: Return is based on a difference in attachment class type. An attachment of class type "Message" will be its MessageID; an attachment of class type "Embedded OLE object" currently returns an error; and, an attachment of class type "File" is path\displayname (if using X00 for the msgID or just DisplayName if using the actual MsgID).

ItemAttachmentSaveAs()

Saves an attachment to a specified directory, or default macros directory.

Token ID

```
AFTKN_ITEM_ATTACHMENT_SAVE_AS or 671
```

Syntax

```
VOID ItemAttachmentSaveAs(ANSISTRING MessageID;
                          WORD Index;
                          ANSISTRING Filename)
```

Parameters

MessageID As ANSISTRING

Unique item identifier, returned by ItemMessageIDFromView().

Index As WORD

Attachment index number (zero-based).

Filename As ANSISTRING

Attachment name. Path is optional.

ItemAttachmentUpdate()

Replaces one attachment with another. It only works on attachments of class type "File."

Token ID

```
AFTKN_ITEM_ATTACHMENT_UPDATE or 753
```

Syntax

```
VOID ItemAttachmentUpdate(ANSISTRING MessageID;
                          WORD Index;
                          ANSISTRING Filename)
```

Parameters

MessageID As ANSISTRING

Unique item identifier, returned by ItemMessageIDFromView().

Index As WORD

Attachment index number (zero-based).

Filename As ANSISTRING

Replacement Attachment.

ItemChangeToAppointment()

Changes the current item to an Appointment item.

Token ID

```
BFTKN_ITEM_CHANGETO_APPT or 435
```

Syntax

```
VOID ItemChangeToAppointment()
```

ItemChangeToMail()

Changes the current item to a Mail item.

Token ID

```
BFTKN_ITEM_CHANGETO_MAIL or 434
```

Syntax

```
VOID ItemChangeToMail()
```

ItemChangeToNote()

Changes the current item to a Note item.

Token ID

```
BFTKN_ITEM_CHANGETO_NOTE or 437
```

Syntax

```
VOID ItemChangeToNote()
```

ItemChangeToPhone()

Changes the current item to a Phone item.

Token ID

```
BFTKN_ITEM_CHANGETO_PHONE or 438
```

Syntax

```
VOID ItemChangeToPhone()
```

ItemChangeToTask()

Changes the current item to a task item.

Token ID

```
BFTKN_ITEM_CHANGETO_TASK or 436
```

Syntax

```
VOID ItemChangeToTask()
```

ItemChecklistMove()

Moves an item in the currently displayed checklist folder to a new position in the list.

Token ID

```
AFTKN_CHECKLIST_MOVE 921
```

Syntax

```
VOID ItemChecklistMove([DWORD IndexFrom];
                       [DWORD IndexTo])
```

Parameters

IndexFrom as DWORD

(Optional) Zero-based index of item to be moved.

IndexTo as DWORD

(Optional) Zero-based index of new position in the list.

ItemChecklistMoveDown()

Moves the currently displayed checklist item down one position in the checklist.

Token ID

```
BFTKN_CHECKLIST_MOVE_DOWN 1086
```

Syntax

```
VOID ItemChecklistMoveDown()
```

ItemChecklistMoveLeft()

Promotes a sub-task in a tasklist such that it becomes a peer to the tasklist item that was previously it’s parent.

Token ID

```
BFTKN_CHECKLIST_MOVE_LEFT or 1122
```

Syntax

```
VOID ItemChecklistMoveLeft()
```

ItemChecklistMoveRight()

Demotes a task item in a tasklist such that it becomes a child of the next item above it that was previously it's peer.

Token ID

```
BFTKN_CHECKLIST_MOVE_RIGHT or 1121
```

Syntax

```
VOID ItemChecklistMoveRight()
```

ItemChecklistMoveToBottom()

Moves the specified checklist item to the bottom of the checklist.

Token ID

```
BFTKN_CHECKLIST_MOVE_TO_BOTTOM 1084
```

Syntax

```
VOID ItemChecklistMoveToBottom([DWORD Index])
```

Parameters

Index as DWORD

(Optional) Zero-based index of the item to move. If not passed in, the currently selected item is moved.

ItemChecklistMoveToTop()

Moves the specified checklist item to the top of the checklist.

Token ID

```
BFTKN_CHECKLIST_MOVE_TO_TOP 1083
```

Syntax

```
VOID ItemChecklistMoveToTop([DWORD Index])
```

Parameters

Index as DWORD

(Optional) Zero-based index of the item to move. If not passed in, the currently selected item is moved.

ItemChecklistMoveUp()

Moves the currently selected checklist item up one position in the checklist.

Token ID

```
BFTKN_CHECKLIST_MOVE_UP 1085
```

Syntax

```
VOID ItemChecklistMoveUp()
```

ItemChecklistNewSubItem()

Create new sub-task as a child of the currently selected task item in a tasklist.

Token ID

```
BFTKN_CHECKLIST_NEW_SUBITEM or 1123
```

Syntax

```
VOID ItemChecklistNewSubItem()
```

ItemComplete()

Has been modified from the old ItemComplete token to accept a Message ID parameter, which allows a one-step completion of the action with no input required from the user. If MessageID is not specified, the token is internally rerouted to the ItemCompleteOpenItem() (page 457) token.

Token ID

```
AFTKN_ITEM_COMPLETE or 918
```

Syntax

```
VOID ItemComplete([ENUM Accept];
                  [ANSISTRING MessageID])
```

Parameters

Accept As ENUM

(Optional) Enumerated values:

1 Done
0 Not Done

MessageID As ANSISTRING

(Optional) Unique item identifier.

ItemCompleteOpenItem()

Has a new mnemonic that was given to the old ItemComplete() (page 456). It changes the Completed state of the currently open item.

Token ID

```
BFTKN_COMPLETE_ITEM or 193
```

Syntax

```
VOID ItemCompleteOpenItem([ENUM Accept])
```

Parameters

Accept As ENUM

(Optional) Enumerated values:

1 Done
0 Not Done

Remarks

When the Accept parameter is not specified, this token toggles the completed state to its other value.

ItemCopyProperties()

Copies selected documents and their properties to another library.

Token ID

```
DTKN_DM_COPY or 98
```

Syntax

```
VOID ItemCopyProperties()
```

ItemCustomReplyDlg()

Displays the reply dialog box to set custom reply options.

Token ID

```
DTKN_CUSTOM_REPLY or 54
```

Syntax

```
VOID ItemCustomReplyDlg()
```

See Also

“ItemReply()” on page 511

ItemDecline()

Declines an appointment, task, or note, and attaches a comment.

Token ID

```
AFTKN_ITEM_DECLINE or 788
```

Syntax

```
VOID ItemDecline([ANSISTRING Comment];
                 [ANSISTRING MessageID];
                 [ENUM AllInstances])
```

Parameters

Comment As ANSISTRING

(Optional) Comment is displayed in an item's information view.

MessageID As ANSISTRING

(Optional) Unique item identifier, returned by ItemMessageIDFromView(). If not specified, declines the selected item.

AllInstances As ENUM

(Optional)

0 No
1 Yes

See Also

“ItemDelete()” on page 465
“ItemDeclineOpenItem()” on page 461

ItemDeclineOpenItem()

Declines a selected or opened appointment, task, or note and attaches a comment.

Token ID

```
BFTKN_REJECT or 195
```

Syntax

```
VOID ItemDeclineOpenItem([ANSISTRING Comment];
                         [BOOLEAN KeepThisItem];
                         [DWORD AcceptLevel])
```

Parameters

Comment As ANSISTRING

(Optional) Comment is displayed in an item's information view.

KeepThisItem As BOOLEAN

(Optional)

AcceptLevel As DWORD

(Optional) Use these values as you would an enumeration (even though this variable is a LONG):

160 FREE
190 TENTATIVE
210 BUSY
251 OUT

See Also

"ItemDeclineWithCommentDlg()" on page 462
"ItemDelete()" on page 465

ItemDeclineWithCommentDlg()

Displays the Decline With Comment dialog box.

Token ID

```
DTKN_REJECT_REPLY or 57
```

Syntax

```
VOID ItemDeclineWithCommentDlg()
```

See Also

“ItemDeclineOpenItem()” on page 461
“ItemDelete()” on page 465

ItemDelegateDlg()

Displays the delegate dialog box.

Token ID

```
DTKN_DELEGATE or 59
```

Syntax

```
VOID ItemDelegateDlg()
```

ItemDelegateOpenItem()

Delegates an appointment, task, note and attaches comments.

Token ID

```
AFTKN_DELEGATE or 607
```

Syntax

```
VOID ItemDelegateOpenItem(ANSISTRING ToUserID;
                          [ANSISTRING RecipComments];
                          [ANSISTRING SenderComments])
```

Parameters

ToUserID As ANSISTRING

Definition (Domain.PostOffice.UserID)

RecipComments As ANSISTRING

(Optional) Definition (Domain.PostOffice.UserID)

SenderComments As ANSISTRING

(Optional) Definition (Domain.PostOffice.UserID)

ItemDelete()

Deletes an item from the Mailbox or Sent Items.

Token ID

```
AFTKN_ITEM_DELETE or 695
```

Syntax

```
VOID ItemDelete([ANSISTRING MessageID];
                [ENUM EmptyItem];
                [DWORD FolderListHandle];
                [ENUM AllInstances];
                [ENUM Retract])
```

Parameters

MessageID As ANSISTRING

(Optional) Unique item identifier, returned by ItemMessageIDFromView(). If not specified, deletes the selected item. If specified, you must also specify EmptyItem.

EmptyItem As ENUM

(Optional) Do not place deleted items in the trash:

0 No (Default)
1 Yes

FolderListHandle As DWORD

(Optional) Handle of a folder list. If not specified, deletes the item from all folders. Handle is returned in a variable by FolderListCreate() or FolderListCreateFromView().

AllInstances As ENUM

(Optional) Deletes all instances of autodate item:

0 No (Default)
1 Yes

Retract As ENUM

(Optional) Retracts a Sent Items item:

292	AllBoxes
291	AllInBoxes
290	ThisOutbox

Remarks

IMPORTANT: EmptyItem ENUM always places any selected or opened Mailbox or Sent Items in the trash. If MessageID is used, UNUM Yes/1 does not put the item in the trash. AllInstances and Retract parameters have problems with selected items in that it prompts or ignores specified parameter enumerations if a MessageID is not used.

ItemDeleteOpenItem()

Deletes a selected or opened Mailbox or Sent Items item.

Token ID

`BFTKN_DELETE_OPEN_ITEM or 209`

Syntax

```
VOID ItemDeleteOpenItem()
```

ItemDigitalSign()

Digitally sign this item.

Token ID

```
BFTKN_DIGITAL_SIGN or 497
```

Syntax

```
VOID ItemDigitalSign(ENUM SignMode)
```

Parameters

SignMode As ENUM

0 No
1 Yes

ItemEncrypt()

Encrypt the entire mail message including all attachments.

Token ID

```
BFTKN_ENCRYPT or 498
```

Syntax

```
VOID ItemEncrypt(ENUM EncryptMode)
```

Parameters

EncryptMode As ENUM

0 No
1 Yes

ItemFolderAltMove()

Moves an item to the folders in a folder list.

Token ID

```
AFTKN_ITEM_FOLDER_ALTMOVE or 767
```

Syntax

```
VOID ItemFolderAltMove(ANSISTRING MessageID;
                       DWORD FolderListHandle)
```

Parameters

MessageID As ANSISTRING

Unique item identifier.

FolderListHandle As DWORD

Handle of the destination folder list.

ItemFolderLink()

Links an item to the folders in a folder list.

Token ID

```
AFTKN_ITEM_FOLDER_LINK or 768
```

Syntax

```
VOID ItemFolderLink(ANSISTRING MessageID;
                    DWORD FolderListHandle)
```

Parameters

MessageID As ANSISTRING

Unique item identifier.

FolderListHandle As DWORD

Handle of the destination folder list.

ItemFolderMove()

Moves an item from one folder list to another.

Token ID

```
AFTKN_ITEM_FOLDER_MOVE or 769
```

Syntax

```
VOID ItemFolderMove(ANSISTRING MessageID;
                    DWORD FolderListHandleDest;
                    DWORD FolderListHandleSrc)
```

Parameters

MessageID As ANSISTRING

Unique item identifier, returned by ItemMessageIDFromView().

FolderListHandleDest As DWORD

Handle of the destination folder list, returned by FolderListCreate().

FolderListHandleSrc As DWORD

Handle of the source folder list, returned by FolderListCreate().

ItemForward()

Forwards a selected or opened Mailbox item to one or more users.

Token ID

```
BFTKN_FORWARD or 274
```

Syntax

```
VOID ItemForward()
```

ItemForwardFlat()

Opens a new Forward Mail window that contains the current open item. This functionality is the same as selecting Forward from the menu (as opposed to selecting Forward as Attachment).

Token ID

```
BFTKN_SIMPLE_FORWARD or 1035
```

Syntax

```
VOID ItemForwardFlat()
```

ItemGetAttribute()

Returns True or False, depending on the state of a specified attribute.

Token ID

```
AFTKN_ITEM_GET_ATTRIBUTE or 673
```

Syntax

```
BOOLEAN ItemGetAttribute(ANSISTRING MessageID;
                         ENUM Attribute)
```

Parameters

MessageID As ANSISTRING

Unique item identifier, returned by ItemMessageIDFromView().

Attribute As ENUM

Specifies the attribute:

Accepted	104
Archived	279
Autodate	289
Completed	138
CustomView	141
Delegated	143
Deleted	144
Forwarding	298
Opened	280
Personal	198
PhoneCalled	0
PhoneCameToSeeYou	5
PhonePleaseCall	1
PhoneReturnValuedYourCall	3
PhoneUrgent	6
PhoneWantsToSeeYou	4
PhoneWillCallAgain	2
Private	200
Read	206

Replying	297
ReplyRequested	207
Resending	299
ReturnReceiptRequested	215
Routed	217
RoutingEndOfLine	303

Return Values

BOOLEAN. True/False.

ItemGetDate()

Returns the date specified by the DateType parameter.

Token ID

```
AFTKN_ITEM_GET_DATE or 674
```

Syntax

```
ANSISTRING ItemGetDate(ANSISTRING MessageID;
                       ENUM DateType;
                       ENUM DateFormatType)
```

Parameters

MessageID As ANSISTRING

Unique item identifier, returned by ItemMessageIDFromView().

DateType As ENUM

Type of date information, as follows:

Constant	Value
AssignedDate	112
BeginDateTime	126
CreateDate	139
DueDate	148
Duration	0
EndDateTime	151

DateFormatType As ENUM

Constant	Value	Comments
Date	176	Returns in format: Month/Day 12HR:MIN AM[PM].
MAPIDate	174	Returns in format: Year/Month/Day 24HR:MIN.

Return Values

ANSISTRING

ItemGetMailboxID()

Returns the mailbox ID of an item's primary recipient.

Token ID

```
AFTKN_ITEM_GET_RECIPIENTID or 773
```

Syntax

```
ANSISTRING ItemGetMailboxID(ANSISTRING MessageID)
```

Parameters

MessageID As ANSISTRING

Unique item identifier, returned by ItemMessageIDFromView.

Return Values

ANSISTRING. Mailbox ID in the form: UserID.PostOffice.Domain.

ItemGetMimeXField()

Retrieve the value of a MIME X-Field associated with a message. X-FIELD’s are set on messages that are delivered to GroupWise by way of the Internet or set manually by the ItemAddMimeXField() call.

Token ID

```
AFTKN_ITEM_GET_MIME_XFIELD or 931
```

Syntax

```
ANSISTRING ItemGetMimeXField(
     ANSISTRING MessageID;
     WORD Index)
```

Parameters

MessageID As ANSISTRING

Unique item identifier, returned by ItemMessageIDFromView().

Index As WORD

The 0-based index of the X-FIELD to retrieve.

Return Values

ANSISTRING. Returns the value of the X-FIELD in “Name: Value” format.

ItemGetMimeXFieldCount()

Retrieve the number of MIME X-FIELD's on an item.

Token ID

```
AFTKN_ITEM_GET_MIME_XFIELD_COUNT or 932
```

Syntax

```
WORD ItemGetMimeXFieldCount(ANSISTRING MessageID)
```

Parameters

MessageID As ANSISTRING

Unique item identifier, returned by ItemMessageIDFromView().

ItemGetOutboxMessageID()

Returns the message ID of a Sent Items item.

Token ID

```
AFTKN_ITEM_GET_OUTBOX_MSGID or 776
```

Syntax

```
ANSISTRING ItemGetOutboxMessageID(ANSISTRING MessageID)
```

Parameters

MessageID As ANSISTRING

Unique item identifier, returned by ItemMessageIDFromView().

Return Values

ANSISTRING. Message ID. ItemGetOutBoxMessageID returns an "undefined function return value" if the MessageID parameter identifies a personal item, or a Mailbox item without a corresponding Sent Items item.

ItemGetPriority()

Returns the priority setting of an item.

Token ID

```
AFTKN_ITEM_GET_PRIORITY or 675
```

Syntax

```
WORD ItemGetPriority(ANSISTRING MessageID)
```

Parameters

MessageID As ANSISTRING

Unique item identifier, returned by ItemMessageIDFromView().

Return Values

WORD. Low 1 Medium 2 High 3

ItemGetReplyToMessageID()

Returns the message ID of the item that is being replied to.

Token ID

```
AFTKN_ITEM_GET_REPLY_MSGID or 814
```

Syntax

```
ANSISTRING ItemGetReplyToMessageID(ANSISTRING MessageID)
```

Parameters

MessageID As ANSISTRING

Message ID of the reply view ("X00").

Return Values

ANSISTRING. Message ID.

ItemGetSecurityClassification()

Returns the security classification for an item as set on the send options tab of a message.

Token ID

```
AFTKN_ITEM_GET_SECURITY_CLASS or 939
```

Syntax

```
ANSISTRING ItemGetSecurityClassification(
     ANSISTRING MessageID)
```

Parameters

MessageID As ANSISTRING

Unique item identifier, returned by ItemMessageIDFromView().

Return Values

ANSISTRING. This will be one of the following string values localized into the language of the running client:

Normal
Proprietary
Confidential
Secret
Top Secret
For Your Eyes Only

ItemGetSenderID()

Returns the user ID of the item’s sender.

Token ID

```
AFTKN_ITEM_GET_SENDERID or 676
```

Syntax

```
ANSISTRING ItemGetSenderID(ANSISTRING MessageID)
```

Parameters

MessageID As ANSISTRING

Unique item identifier, returned by ItemMessageIDFromView.

Return Values

ANSISTRING. A user ID in the form Domain.PostOffice(UserID).

ItemGetSource()

Returns the location of a specified item.

Token ID

```
AFTKN_ITEM_GET_SOURCE or 677
```

Syntax

```
WORD ItemGetSource(ANSISTRING MessageID)
```

Parameters

MessageID As ANSISTRING

Unique item identifier, returned by ItemMessageIDFromView().

Return Values

WORD. Command fails on encapsulated items.
Mailbox 1
Sent Items 2
Personal 3

ItemGetText()

Returns the contents of a text field (edit box).

Token ID

```
AFTKN_ITEM_GET_TEXT or 678
```

Syntax

```
ANSISTRING ItemGetText(ANSISTRING MessageID;
                       ENUM Field)
```

Parameters

MessageID As ANSISTRING

Unique item identifier, returned by ItemMessageIDFromView().

Field As ENUM

Specifies the field, as follows:

Constant	Value	Comments
Authority	4	
BC	3	Only retrieves blind copy information from a Sent Items when you are the sender or have proxy rights.
Caller	6	
CC	2	
Company	7	
From	1	
Message	10	
Phone	8	
Place	5	
Subject	9	
TaskCategory	227	
TaskPriority	230	
To	0	
ViewName	267	

Return Values

ANSISTRING. String contents of a text field, or empty string if the text field is empty.

Remarks

The message body field (in the Message View and the Object API) has a limit of 32,000 characters. The Message View will stop accepting data after 32,000 characters.

ItemGetType()

Returns an item's type.

Token ID

```
AFTKN_ITEM_GET_TYPE or 679
```

Syntax

```
WORD ItemGetType(ANSISTRING MessageID)
```

Parameters

MessageID As ANSISTRING

Unique item identifier, returned by ItemMessageIDFromView().

Return Values

WORD
- Appointment 1
- Mail 2
- Note 3
- Phone Message 4
- Task 5

Remarks

The message body field (in the Message View and the Object API) has a limit of 32,000 characters. The Message View will stop accepting data after 32,000 characters.

See Also

"ItemInfoOpenItem()" on page 491

ItemInfo()

Displays the properties for the item identified by MessageID.

Token ID

```
AFTKN_ITEM_INFO or 790
```

Syntax

```
VOID ItemInfo(ANSISTRING MessageID)
```

Parameters

MessageID As ANSISTRING

ID of the given message.

ItemInfoOpenItem()

Displays the item Information view of a selected item.

Token ID

```
BFTKN_INFO or 273
```

Syntax

```
VOID ItemInfoOpenItem()
```

ItemIsValid()

Returns True if an item is valid, False if not. An item is invalid after it is deleted, or when it is associated with an unproxied user.

Token ID

```
AFTKN_ITEM_MSGID_ISVALID or 681
```

Syntax

```
BOOLEAN ItemIsValid(ANSISTRING MessageID)
```

Parameters

MessageID As ANSISTRING

Unique item identifier, returned by ItemMessageIDFromView().

Return Values

BOOLEAN. True/False.

ItemListCreate()

Creates an item list from a current filter or folder list.

Token ID

```
AFTKN_ITEMLIST_CREATE or 696
```

Syntax

```
DWORD ItemListCreate([ANSISTRING UserID];
                     [DWORD FilterHandle];
                     [DWORD FolderListHandle])
```

Parameters

UserID As ANSISTRING

(Optional) User network ID.

FilterHandle As DWORD

(Optional) Filter list handle.

FolderListHandle As DWORD

(Optional) Folder list handle.

Return Values

DWORD. Item list handle.

IMPORTANT: To avoid losing system resources, make sure you delete an item list handle when it is no longer needed.

See Also

"ItemListDelete()" on page 496

ItemListCreateFromControl()

Creates an item list from a control, such as an Appointment list.

Token ID

```
AFTKN_ITEMLIST_CREATE_FM_CTRL or 697
```

Syntax

```
DWORD ItemListCreateFromControl(ENUM ControlListType;
                                [ENUM SelectedItemsOnly];
                                [ANSISTRING ControlName])
```

Parameters

ControlListType As ENUM

Specifies the control list type as follows:

Constant	Value	Comments
AppointmentList	108	Apply to calendar views.
ItemList	44	Apply to Mailbox, Sent Items, or Trash views.
NoteList	188	Apply to calendar views.
SelectedControl	219	Apply to all item views.
TaskList	228	Apply to calendar views.
WeekControl	268	Apply to a Week Calendar view.

SelectedItemsOnly As ENUM

(Optional) Specifies whether to use selected items only:

Constant	Value	Comments
No	0	Default
Yes	1	If specified, the index parameter of the ItemListGetItem command must be 0.

ControlName As ANSISTRING

(Optional) Custom control name.

Return Values

DWORD. Item list handle.

IMPORTANT: To avoid losing system resources, make sure you delete an item list handle when it is no longer needed.

See Also

“ItemListDelete()” on page 496

ItemListDelete()

Deletes an item list handle from memory.

Token ID

```
AFTKN_ITEMLIST_DELETE or 698
```

Syntax

```
VOID ItemListDelete(DWORD Handle)
```

Parameters

Handle As DWORD

Item list handle.

See Also

“ItemListCreate()” on page 493
“ItemListCreateFromControl()” on page 494

ItemListGetCount()

Returns the number of items in a list.

Token ID

```
AFTKN_ITEMLIST_GET_COUNT or 699
```

Syntax

```
WORD ItemListGetCount(DWORD Handle)
```

Parameters

Handle As DWORD

Item list handle.

Return Values

WORD. Number of list items.

ItemListGetItem()

Returns the message ID of a list item. ItemListCreate or ItemListCreateFromControl creates a list and returns an Item list handle.

Token ID

```
AFTKN_ITEMLIST_GET_MSGID or 700
```

Syntax

```
ANSISTRING ItemListGetItem(DWORD Handle;
                           WORD Index)
```

Parameters

Handle As DWORD

Item list handle.

Index As WORD

Index number (zero-based) of a list item.

Return Values

ANSISTRING. Message string.

ItemMarkPrivate()

Marks private the opened item or all selected items.

Token ID

```
BFTKN_MARK_PRIVATE or 336
```

Syntax

```
VOID ItemMarkPrivate(ENUM Private)
```

Parameters

Private As ENUM

If no parameter is specified, acts as a toggle.

Constant	Value	Comments
No	0	Unmarks an item.
Yes	1	

Index number (zero-based) of a list item.

ItemMessageIDFromView()

Returns the message ID of an active view.

Token ID

```
AFTKN_ITEM_MSGID_FROM_VIEW or 672
```

Syntax

```
ANSISTRING ItemMessageIDFromView()
```

Return Values

ANSISTRING. Message ID. If the item is being created, the message ID is always "X00".

ItemMoveToChecklistFolder()

Moves the currently selected checklist item(s) to the checklist folder.

Token ID

```
BFTKN_CHECKLIST_MOVE_TO_FOLDER 1081
```

Syntax

```
VOID ItemMoveToChecklistFolder()
```

ItemNewPropertySheet()

Creates a new document.

Token ID

```
DTKN_DM_NEW or 99
```

Syntax

```
VOID ItemNewPropertySheet()
```

ItemNextTabSelect()

Selects the next tab in a tabbed item view.

Token ID

```
BFTKN_NEXT_TAB_SELECT 1155
```

Syntax

```
VOID ItemNextTabSelect()
```

ItemOpen()

Opens an item with a corresponding MessageID (see MessageID parameter).

Token ID

```
AFTKN_ITEM_OPEN or 682
```

Syntax

```
VOID ItemOpen(ANSISTRING MessageID;
              [ANSISTRING UserID])
```

Parameters

MessageID As ANSISTRING

Unique item identifier, returned by ItemMessageIDFromView().

UserID As ANSISTRING

(Optional) User ID. This parameter permits opening messages on a proxiable mailbox.

ItemPost()

Posts the current item.

Token ID

```
BFTKN_POST 1111
```

Syntax

```
VOID ItemPost()
```

ItemPrint()

Prints the item specified by MessageID. Brings up the print dialog just as if you clicked File | Print or the Print toolbar button.

Token ID

```
AFTKN_ITEM_PRINT or 872
```

Syntax

```
VOID ItemPrint(ANSISTRING MessageID)
```

Parameters

MessageID As ANSISTRING

Unique item identifier, returned by ItemMessageIDFromView().

ItemRead()

Opens a selected item to read.

Token ID

```
BFTKN_READ or 272
```

Syntax

```
VOID ItemRead()
```

See Also

“ItemReadNext()” on page 509
“ItemReadPrevious()” on page 510

ItemReadLater()

Changes selected items to appear unopened in the GroupWise client.

Token ID

`BFTKN_READ_LATER or 352`

Syntax

```
VOID ItemReadLater()
```

See Also

“ItemReadNext()” on page 509
“ItemReadPrevious()” on page 510

ItemReadNext()

Opens an item after an already opened item.

Token ID

```
BFTKN_READ_NEXT or 324
```

Syntax

```
VOID ItemReadNext()
```

See Also

“ItemRead()” on page 507
“ItemReadPrevious()” on page 510

ItemReadPrevious()

Opens the item before an already opened item.

Token ID

```
BFTKN_READ_PREV or 325
```

Syntax

```
VOID ItemReadPrevious()
```

See Also

“ItemRead()” on page 507
“ItemReadNext()” on page 509

ItemReply()

Is modified from the old ItemReply token to accept a MessageID parameter that allows a one-step completion of the action with no input from the user. If MessageID is not specified, the token is internally rerouted to the ItemReplyOpenItem token.

Token ID

```
AFTKN_ITEM_REPLY or 919
```

Syntax

```
VOID ItemReply([ENUM ReplyTo];
               [ENUM IncludeText];
               [ANSISTRING MessageID])
```

Parameters

ReplyTo As ENUM

(Optional) Enumerated values:

Constant	Value
All	7
AllPrivate	39
Sender	1
SenderPrivate	33
Discussion	16
DiscussionNNTP	144
DiscussionORIG	256

IncludeText As ENUM

(Optional) Enumerated values:

Constant	Value
No (Default)	0
Yes	1

MessageID As ANSISTRING

(Optional) Unique message identifier.

ItemReplyOpenItem()

Is the new mnemonic given to the old ItemReply() (page 511) token. It creates a new reply to the message that is currently open.

Token ID

```
BFTKN_REPLY or 271
```

Syntax

```
VOID ItemReplyOpenItem([ENUM ReplyTo];
                       [ENUM IncludeText])
```

Parameters

ReplyTo As ENUM

(Optional) Enumerated values:

Constant	Value
All	7
AllPrivate	39
Sender	1
SenderPrivate	33
Discussion	16
DiscussionNNTP	144
DiscussionORIG	256

IncludeText As ENUM

(Optional) Enumerated values:

0 No
1 Yes

Remarks

ItemReplyOpenItem does not actually send the reply. It opens a new reply window into which the user can compose the reply message.

ItemResend()

Resends a selected or opened Sent Items item.

Token ID

```
BFTKN_RESEND or 275
```

Syntax

```
VOID ItemResend([ENUM RetractOriginal])
```

Parameters

RetractOriginal As ENUM

(Optional) Specifies whether to prompt if not specified.

Constant	Value
No	0
Yes	1

ItemRoute()

Routes a message task to different users in turn.

Token ID

```
BFTKN_TOGGLE_ROUTING or 192
```

Syntax

```
VOID ItemRoute([ENUM DoRouting])
```

Parameters

DoRouting As ENUM

(Optional) Turn on routing. If no parameter is specified, acts as a toggle.

Constant	Value
No	0
Yes	1

See Also

“ItemResend()” on page 513

ItemSaveInfo()

Saves item information (recipient names, routes, file sizes, creation date, and security options) to a specified file. ItemSaveInfo saves the properties of received messages and does not work for draft messages.

Token ID

```
AFTKN_ITEM_INFO_SAVE or 680
```

Syntax

```
VOID ItemSaveInfo(ANSISTRING MessageID;
                  ANSISTRING Filename;
                  ENUM FileFormat)
```

Parameters

MessageID As ANSISTRING

Unique item identifier, returned by ItemMessageIDFromView.

Filename As ANSISTRING

Path and name of a file.

FileFormat As ENUM

Specifies the file format as follows:

Constant	Value
Ansitext	106
WordPerfect60	269

ItemSaveMessage()

Saves the information contained in an item view to a specified file.

Token ID

```
AFTKN_ITEM_SAVE_MESSAGE or 683
```

Syntax

```
VOID ItemSaveMessage(ANSISTRING MessageID;
                     ANSISTRING Filename;
                     ENUM FileFormat)
```

Parameters

MessageID As ANSISTRING

Unique item identifier, returned by ItemMessageIDFromView() (page 500).

Filename As ANSISTRING

Full path and name of a file.

FileFormat As ENUM

Specifies the file format as follows:

Constant	Value
Ansitext	106
WordPerfect60	269
Mime	900
RichText	899

ItemSaveMessageDlg()

Displays the Save Message dialog box of an item view, or the Save Message Entries dialog box of an Mailbox, Sent Items, or Calendar view (selected or opened).

Token ID

```
DTKN_SAVE_DLG or 71
```

Syntax

```
VOID ItemSaveMessageDlg()
```

ItemSaveMessageDraft()

Saves a draft copy of the current message in the specified folder. If no folder is specified, the user is prompted to indicate which folder to use.

Token ID

```
AFTKN_ITEM_SAVE_MESSAGE_DRAFT or 915
```

Syntax

```
VOID ItemSaveMessageDraft([ANSISTRING FolderPath])
```

Parameters

FolderPath As ANSISTRING

The fully qualified path to the folder in which the message draft should be saved.

ItemSaveView()

Saves a custom view to a file.

Token ID

```
AFTKN_ITEM_SAVE_VIEW or 684
```

Syntax

```
VOID ItemSaveView(ANSISTRING MessageID;
                  ANSISTRING Filename)
```

Parameters

MessageID As ANSISTRING

Unique item identifier, returned by ItemMessageIDFromView().

Filename As ANSISTRING

Full path and name of a file.

ItemSaveViewDlg()

Displays the Save View dialog box.

Token ID

```
DTKN_SAVE_AS_DLG or 70
```

Syntax

```
VOID ItemSaveViewDlg()
```

ItemSend()

Sends a new item. The item must be open.

Token ID

```
BFTKN_SEND or 266
```

Syntax

```
DWORD ItemSend([ENUM AutoSpellCheck];
               [ANSISTRING AccountName])
```

Parameters

AutoSpellCheck As ENUM

(Optional) Enumerated values:

0 No
1 Yes

AccountName As ANSISTRING

(Optional) Name of the account.

Return Values

DWORD

ItemSetAlarm()

Sets an alarm for the appointment specified by MessageID. This command fails if the item identified by MessageID is not an appointment or it the appointment is past.

Token ID

```
AFTKN_ITEM_SET_ALARM or 655
```

Syntax

```
VOID ItemSetAlarm(ANSISTRING MessageID;
                  WORD HoursBefore;
                  WORD MinutesBefore;
                  [ANSISTRING ProgramToLaunch])
```

Parameters

MessageID As ANSISTRING

Unique item identifier, returned by ItemMessageIDFromView().

HoursBefore As WORD

Hours before the appointment to sound the alarm.

MinutesBefore As WORD

Minutes before the appointment to source the alarm.

ProgramToLaunch As ANSISTRING

(Optional) Path and name of a program to launch when the alarm goes off.

ItemSetAllDayEvent()

Marks an event as an all day event.

Token ID

```
BFTKN_SET_ALLDAYEVENT or 1186
```

Syntax

```
VOID ItemSetAllDayEvent([ENUM AllDay])
```

Parameters

AllDay As ENUM

(Optional)

0 No
1 Yes

ItemSetAttribute()

Sets an attribute for the item specified by MessageID.

Token ID

```
AFTKN_ITEM_SET_ATTRIBUTE or 685
```

Syntax

```
VOID ItemSetAttribute(ANSISTRING MessageID;
                      ENUM ModifyAttribute;
                      ENUM AttributeValue)
```

Parameters

MessageID As ANSISTRING

Unique item identifier, returned by ItemMessageIDFromView().

ModifyAttribute As ENUM

Apply attributes to specified item conditions The conditions are: creating personal item, creating group item, existing personal item, and existing group item:

104	Accepted
138	Completed
200	Private
206	Read

AttributeValue As ENUM

Set Attribute:

0 No
1 Yes

Remarks

You can only set attributes under certain conditions. For example, you can accept or decline scheduled items (appointments, tasks and notes), but not mail or phone items. When using this command, correctly match attributes and conditions.

ItemSetDate()

Sets the start and/or end date, or duration of Tasks, Appointments and Notes. EndMinute is not optional when trying to set the Due On date of a Task.

Token ID

```
AFTKN_ITEM_SET_DATE or 792
```

Syntax

```
VOID ItemSetDate(ANSISTRING MessageID;
                 [WORD StartDay];
                 [WORD StartMonth];
                 [WORD StartYear];
                 [WORD StartMinute];
                 [WORD StartHour];
                 [WORD EndDay];
                 [WORD EndMonth];
                 [WORD EndYear];
                 [WORD EndMinute];
                 [WORD EndHour];
                 [WORD DurationMinutes];
                 [WORD DurationHours])
```

Parameters

MessageID As ANSISTRING

StartDay As WORD

(Optional)

StartMonth As WORD

(Optional)

StartYear As WORD

(Optional)

StartMinute As WORD

(Optional)

StartHour As WORD

(Optional)

EndDay As WORD

(Optional)

EndMonth As WORD

(Optional)

EndYear As WORD

(Optional)

EndMinute As WORD

(Optional)

EndHour As WORD

(Optional)

DurationMinutes As WORD

(Optional)

DurationHours As WORD

(Optional)

ItemSetItemType()

Specifies an item type for a new item.

Token ID

```
AFTKN_ITEM_SET_TYPE or 688
```

Syntax

```
VOID ItemSetItemType(ANSISTRING MessageID;
                     ENUM ItemType;
                     ENUM IsPersonal)
```

Parameters

MessageID As ANSISTRING

Unique item identifier, returned by ItemMessageIDFromView().

ItemType As ENUM

New item type:

274	Appointment
276	Mail
278	Note
8	Phone
505	Profile
506	Search
277	Task

IsPersonal As ENUM

0 No
1 Yes

ItemSetPriority()

Assign priority to a new item.

Token ID

```
AFTKN_ITEM_SET_PRIORITY or 686
```

Syntax

```
VOID ItemSetPriority(ANSISTRING MessageID;
                     ENUM Priority)
```

Parameters

MessageID As ANSISTRING

Unique item identifier, returned by ItemMessageIDFromView().

Priority As ENUM

Specifies the priority:

0	Low
1	Normal
2	High

ItemSetText()

Appends to, or replaces the text of, a specified field.

Token ID

```
AFTKN_ITEM_SET_TEXT or 687
```

Syntax

```
VOID ItemSetText(ANSISTRING MessageID;
                 ENUM Field;
                 ANSISTRING FieldText;
                 [ENUM AppendText];
                 [ENUM ClearChangeNotification])
```

Parameters

MessageID As ANSISTRING

Unique item identifier, returned by ItemMessageIDFromView().

Field As ENUM

Enumerated values:

4	Authority
3	BC
6	Caller
2	CC
7	Company
1	From
10	Message
8	Phone
5	Place
9	Subject
227	TaskCategory
230	TaskPriority
0	To
267	ViewName

FieldText As ANSISTRING

Replacement text.

AppendText As ENUM

(Optional) Enumerated values:

0 No
1 Yes

ClearChangeNotification As ENUM

(Optional) Enumerated values:

0 No
1 Yes

Remarks

The message body field (in the message view and the Object API) has a limit of 32,000 characters and stops accepting data after 32,000 characters.

Example

The following example changes the subject text of a new message (as denoted by X00). Both the MessageID and FieldText need to be surrounded by double quotes.

```
ItemSetText("X00"; 9; "This is a new subject");
```

ItemShowInChecklist()

Displays the currently selected items in the checklist folder.

Token ID

```
BFTKN_CHECKLIST_SHOW_IN 1082
```

Syntax

```
VOID ItemShowInChecklist([ENUM Show])
```

Parameters

Show as ENUM

(Optional) Specifies the following:

0 No
1 Yes

ItemUndelete()

Restores an item from Trash (unless the item view has focus or is on top). The Mailbox or main window must be on top.

Token ID

```
AFTKN_ITEM_UNDELETE or 791
```

Syntax

```
VOID ItemUndelete([ANSISTRING MessageID])
```

Parameters

MessageID As ANSISTRING

(Optional) Unique item identifier, returned by ItemMessageIDFromView().

See Also

“ItemUndeleteOpenItem()” on page 533

ItemUndeleteOpenItem()

Restores a selected item from Trash. Will not work if item view has focus or is on top. Mailbox or main window must be on top.

Token ID

```
BFTKN_UNDELETE or 191
```

Syntax

```
VOID ItemUndeleteOpenItem()
```

See Also

“ItemUndelete()” on page 532

ItemWaitForClose()

This function will block until an item view is sent or closed by the user.

Token ID

`AFTKN_ITEM_WAIT_FOR_CLOSE or 952`

Syntax

```
BOOL ItemWaitForClose()
```

Return Values

Returns TRUE if the item was successfully sent. Returns FALSE if the item was closed or cancelled.

JMAddUserToPAB()

Adds the sender address of the selected item (or open item) to the frequent contacts list of the Personal Address Book (PAB).

Token ID

```
BFTKN_JM_ADDTOPAB 1124
```

Syntax

```
VOID JMAddUserToPAB()
```

Remarks

The selected (or open) item must be a candidate for junk mail handling (delivered through GWIA) and must be in the trash folder or junk mail folder.

JMBlockSender()

Invokes the Block Sender dialog for the selected items.

Token ID

```
BFTKN_JM_BLOCK_SENDER 1130
```

Syntax

```
VOID JMBlockSender()
```

Remarks

At least one of the selected items must be a candidate for junk mail handling (and be delivered through GWIA).

JMJunkSender()

Invokes the Junk Sender dialog for the selected items.

Token ID

```
BFTKN_JM_JUNK_SENDER 1129
```

Syntax

```
VOID JMJunkSender()
```

Remarks

At least one of the selected items must be a candidate for junk mail handling (and be delivered through GWIA).

JMTrustSender()

Invokes the Trust Sender dialog for the selected items.

Token ID

```
BFTKN_JM_TRUST_SENDER 1128
```

Syntax

```
VOID JMTrustSender()
```

Remarks

At least one of the selected items must be a candidate for junk mail handling (and be delivered through GWIA).

JunkMailHandling()

Invokes the junk mail handling dialog.

Token ID

```
DTKN_JUNK_MAIL 137
```

Syntax

```
VOID JunkMailHandling()
```

JunkMailSettings()

Modifies the settings for junk mail handling.

Token ID

```
AFTKN_JUNKMAIL_SETTINGS 922
```

Syntax

```
VOID JunkMailSettings([BOOLEAN BlockListEnabled];
                      [BOOLEAN JunkListEnabled];
                      [BOOLEAN JunkPabEnabled];
                      [ANSISTRING JunkFolderName];
                      [BOOLEAN JunkFolderAutoRemove];
                         [DWORD JunkFolderRetention])
```

Parameters

BlockListEnabled as BOOLEAN

(Optional)

JunkListEnabled as BOOLEAN

(Optional)

JunkPabEnabled as BOOLEAN

(Optional)

JunkFolderName as ANSISTRING

(Optional) Do not use. This parameter is disabled.

JunkFolderAutoRemove as BOOLEAN

(Optional)

JunkFolderRetention as DWORD

(Optional)

6 Tokens Reference (L-Z)

Tokens are low-level events that are packaged into a meaningful representation and are subscribed or published by one of the Token functions (see “Subscribe and Publish” on page 22.)

Information on each token is contained in the following alphabetical sections:

For more token information, see Chapter 5, “Tokens Reference (A-K),” on page 61.

L-N

This section contains information about the following tokens:

ListViewAddressCards()

View the current item list of contacts as address cards.

Token ID

```
BFTKN_LVS_ADDRESSCARD or 1245
```

Syntax

```
VOID ListViewAddressCards([ENUM State])
```

Parameters

State As ENUM

(Optional) If not specified, than the setting is toggled

0 Off
1 On

Remarks

Only lists of contacts can be viewed as address cards.

ListViewCalendar()

Allows only the calendar items in the selected folder to be displayed.

Token ID

```
BFTKN_VIEWBY_PROJECT_CAL or 349
```

Syntax

```
VOID ListViewCalendar([ENUM State])
```

Parameters

State As ENUM

(Optional) If not specified, acts as a toggle:

0 Off!
1 On!

ListViewChecklist()

Displays the current folder in checklist mode.

Token ID

```
BFTKN_LVS_CHECKLIST 1080
```

Syntax

```
VOID ListViewChecklist()
```

ListViewColumns()

Displays the Item List using columns.

Token ID

```
BFTKN_LVS_COLUMNS or 1186
```

Syntax

```
VOID ListViewColumns([ENUM State])
```

Parameters

State As ENUM

(Optional)

0 No
1 Yes

ListViewDetails()

Displays the Item List, using details.

Token ID

```
BFTKN_LVS_DETAILS or 341
```

Syntax

```
VOID ListViewDetails([ENUM State])
```

Parameters

State As ENUM

(Optional) If not specified, acts as a toggle:

0 Off!
1 On!

ListViewMessagePreview()

Display a two line preview of the message body in the item list.

Token ID

```
BFTKN_LVS_MSG_PREVIEW or 1222
```

Syntax

```
VOID ListViewMessagePreview([ENUM State])
```

Parameters

State As ENUM

(Optional) If not specified, then the setting is toggled.

0 Off
1 On

ListViewMessageThread()

Displays incoming and outgoing messaging in discussion format.

Token ID

```
BFTKN_LVS_MSG_THREAD or 354
```

Syntax

```
VOID ListViewMessageThread([ENUM State])
```

Parameters

State As ENUM

(Optional) If not specified, acts as a toggle:

0 Off!
1 On!

ListViewPanels()

Displays items grouped into panels.

Token ID

```
BFTKN_LVS_COLUMNS or 1186
```

Syntax

```
VOID ListViewPanels([ENUM State])
```

Parameters

State As ENUM

(Optional)

0 No
1 Yes

ListViewShowGroupHeaders()

Displays items and group labels that are grouped by the current sort key.

Token ID

```
BFTKN_LVS_SHOW_GROUP_HEADERS or 1167
```

Syntax

```
VOID ListViewShowGroupHeaders()
```

ListViewSummary()

Displays summary items (without using columns).

Token ID

```
BFTKN_LVS_SUMMARY or 1166
```

Syntax

```
VOID ListViewSummary([ENUM State])
```

Parameters

State As ENUM

(Optional)

0 No
1 Yes

MainWindowShow()

Shows the main window when it is hidden, and activates the main window if it is behind an item view (not displayed), or maximizes the main window if it is minimized.

Token ID

```
BFTKN_OPEN_MAINWND or 301
```

Syntax

```
VOID MainWindowShow()
```

ManageCalendarsDlg()

Displays the Calendar Management dialog box to create, delete, rename, or pick a color for your calendars.

Token ID

```
DTKN_MANAGE_CALENDARS or 140
```

Syntax

```
VOID ManageCalendarsDlg()
```

MarkOfficialVersion()

Marks the selected document version as the Official Version.

Token ID

```
BFTKN_DM_MARK_OFFICIAL_VER or 428
```

Syntax

```
VOID MarkOfficialVersion()
```

MarkRead()

Marks the selected items as read.

Token ID

```
BFTKN_MARK_READ or 477
```

Syntax

```
VOID MarkRead()
```

MessengerLaunch()

Launches GroupWise Messenger.

Token ID

`BFTKN_IM_LAUNCH or 1018`

Syntax

```
VOID MessengerLaunch()
```

MessengerSendIM()

Sends a new instant message from GroupWise Messenger.

Token ID

```
BFTKN_IM_SEND_IM 1021
```

Syntax

```
VOID MessengerSendIM()
```

Remarks

GroupWise Messenger must be running.

MessengerShowContacts()

Displays the GroupWise Messenger contact list.

Token ID

```
BFTKN_IM_SHOW_BUDDY_LIST 1022
```

Syntax

```
VOID MessengerShowContacts()
```

Remarks

GroupWise Messenger must be running.

MessengerShowPreferences()

Displays the GroupWise Messenger options dialog.

Token ID

```
BFTKN_IM_PREFERENCES 1019
```

Syntax

```
VOID MessengerShowPreferences()
```

Remarks

GroupWise Messenger must be running.

MessengerSignOff()

Logs out of GroupWise Messenger.

Token ID

```
BFTKN_IM_SIGNOFF or 1020
```

Syntax

```
VOID MessengerSignOff()
```

ModeCaching()

Puts the client into Caching mode.

Token ID

```
BFTKN_MODE_CACHE or 1009
```

Syntax

```
VOID ModeCaching()
```

ModeOffline()

Places the client in offline mode.

Token ID

```
BFTKN_MODE_OFFLINE or 1157
```

Syntax

```
VOID ModeOffline()
```

ModeOnline()

Puts the client into Online mode.

Token ID

```
BFTKN_MODE_MASTER or 1008
```

Syntax

```
VOID ModeOnline()
```

ModeRemote()

Puts the client into Remote mode.

Token ID

```
BFTKN_MODE_REMOTE or 1010
```

Syntax

```
VOID ModeRemote()
```

ModifyDistributionList()

Displays a dialog allowing the user to modify the distribution list of an appointment that they sent.

Token ID

`BFTKN_MODIFY_DISTLIST or 1242`

Syntax

```
VOID ModifyDistributionList()
```

ModifyVacationRule()

Modifies a user's vacation rule.

Token ID

```
BFTKN_MODIFY_VACATION_RULE or 1251
```

Syntax

```
VOID ModifyVacationRule(
     ANSISTRING Name;
     ANSISTRING Subject;
     ANSISTRING Message;
     BOOL IsActive;
     [DWORD] StartDate;
     [DWORD] EndDate;
     BOOL MarkBusy;
     BOOL ReplyToExternalUsers)
```

Parameters

Name As ANSISTRING

The name of the vacation rule. This may be obtained by calling FindVacationRule().

Subject As ANSISTRING

The subject of the email message to send in response to email that arrives while the user is on vacation.

Message As ANSISTRING

The message body of the email message to send in response to email that arrives while the user is on vacation.

IsActive As BOOL

TRUE to activate rule, FALSE to deactivate it.

StartDate As DWORD

(Optional) The date from which the rule will be active. If this is not set then the rule is active immediately until it is manually disabled.

EndDate As DWORD

(Optional) The date from which the rule is active. If this is not set then the rule is active immediately until it is manually disabled.

MarkBusy As BOOL

Set to TRUE to automatically create an All-Day-Event that will block the user's schedule for the duration of the time that the user is on vacation.

ReplyToExternalUsers As BOOL

TRUE indicates that a vacation response should be sent to external users. FALSE indicates that vacation response should only be sent to users on the same GroupWise system.

MoviePause()

Pauses a multi-media graphic running in the graphics box of a customized view.

Token ID

```
BFTKN_PAUSEMOVIE or 186
```

Syntax

```
VOID MoviePause()
```

Remarks

This token works only after you manually pause the AVI file by right-clicking and choosing Pause from the menu.

See Also

“MovieResume()” on page 570

MovieResume()

Resumes play of a multi-media graphic, paused in the graphics box of a customized view.

Token ID

```
BFTKN_RESUMEMOVIE or 188
```

Syntax

```
VOID MovieResume()
```

See Also

“MoviePause()” on page 569

MultiUserList()

Brings up the multi user calendar dialog.

Token ID

```
DTKN_MULTIUSER_LIST or 124
```

Syntax

```
VOID MultiUserList()
```

NewAppointment()

Creates a new appointment item.

Token ID

```
BFTKN_SEND_STDAPPT or 182
```

Syntax

```
VOID NewAppointment()
```

NewAppointmentToContacts()

Opens a new appointment item addressed to the selected contacts.

Token ID

```
BFTKN_SEND_CONTACT_STDAPPT 1150
```

Syntax

```
VOID NewAppointmentToContacts()
```

NewContact()

Invokes the New Contact dialog.

Token ID

```
BFTKN_NEW_CONTACT 1107
```

Syntax

```
VOID NewContact()
```

NewGroup()

Invokes the New Group dialog.

Token ID

```
BFTKN_NEW_GROUP 1110
```

Syntax

```
VOID NewGroup()
```

NewMail()

Creates a new mail item.

Token ID

```
BFTKN_SEND_STDMAIL or 167
```

Syntax

```
VOID NewMail()
```

NewMailToContacts()

Opens a new mail item addressed to the selected contacts.

Token ID

```
BFTKN_SEND_CONTACT_STDMAIL 1149
```

Syntax

```
VOID NewMailToContacts()
```

NewNote()

Creates a new note item.

Token ID

```
BFTKN_SEND_STDNOTE or 181
```

Syntax

```
VOID NewNote()
```

NewNoteToContacts()

Opens a new note item addressed to the selected contacts.

Token ID

```
BFTKN_SEND_CONTACT_STDNOTE 1152
```

Syntax

```
VOID NewNoteToContacts()
```

NewOrganization()

Invokes the New Organization dialog.

Token ID

```
BFTKN_NEW_ORGANIZATION 1109
```

Syntax

```
VOID NewOrganization()
```

NewPhone()

Creates a new phone message item.

Token ID

```
BFTKN_SEND_STDPHON or 196
```

Syntax

```
VOID NewPhone()
```

NewPhoneToContacts()

Opens a new phone item addressed to the selected contacts.

Token ID

```
BFTKN_SEND_CONTACT_STDPHON 1153
```

Syntax

```
VOID NewPhoneToContacts()
```

NewResource()

Invokes the New Resource dialog.

Token ID

```
BFTKN_NEW_RESOURCE 1108
```

Syntax

```
VOID NewResource()
```

NewsGroupsNNTPDlg()

Brings up a "subscribe to a newsgroup" dialog.

Token ID

```
DTKN_NEWS_GROUPS_NNTP or 133
```

Syntax

```
VOID NewsGroupsNNTPDlg()
```

Remarks

The NNTP account you are interested in should be previously selected in the folder tree.

NewTask()

Creates a new task item.

Token ID

```
BFTKN_SEND_STDTODO or 180
```

Syntax

```
VOID NewTask()
```

NewTaskToContacts()

Opens a new task item addressed to the selected contacts.

Token ID

```
BFTKN_SEND_CONTACT_STDTODO 1151
```

Syntax

```
VOID NewTaskToContacts()
```

NewTopic()

Creates a new topic item. This token is referred to in GroupWise as a new discussion (File > New > Discussion).

Token ID

```
BFTKN_SEND_SHRFOLD or 262
```

Syntax

```
VOID NewTopic()
```

NextThread()

Moves the cursor to the next thread.

Token ID

```
BFTKN_NEXT_THREAD or 388
```

Syntax

```
VOID NextThread()
```

NextUnreadItem()

Moves the cursor to the next unread item.

Token ID

```
BFTKN_NEXT_UNREAD or 390
```

Syntax

```
VOID NextUnreadItem()
```

NNTPCancel()

Attempts to cancel (remove from the server) the selected NNTP item.

Token ID

```
BFTKN_NNTP_CANCEL or 1053
```

Syntax

```
VOID NNTPCancel()
```

Remarks

If you do not have cancel rights, the cancel attempt may be rejected by the server.

NNTPCollapseAll()

Collapses all threads in a newsgroup.

Token ID

```
BFTKN_NNTP_COLLAPSE_ALL or 1016
```

Syntax

```
VOID NNTPCollapseAll([ANSISTRING FolderName])
```

Parameters

FolderName As ANSISTRING

(Optional) NNTP newsgroup on which to operate.

NNTPExpandAll()

Expands all threads in a newsgroup.

Token ID

```
BFTKN_NNTP_EXPAND_ALL or 1045
```

Syntax

```
VOID NNTPExpandAll([ANSISTRING FolderName])
```

Parameters

FolderName As ANSISTRING

(Optional) NNTP newsgroup on which to operate.

NNTPMarkAllRead()

Marks every item in a newsgroup to read.

Token ID

```
BFTKN_NNTP_MARK_ALL_READ or 1017
```

Syntax

```
VOID NNTPExpandAll([ANSISTRING FolderName])
```

Parameters

FolderName As ANSISTRING

(Optional) NNTP newsgroup on which to operate.

NNTPSearch()

Invokes a dialog to search the currently selected newsgroup.

Token ID

```
BFTKN_SEARCH_NNTP or 1011
```

Syntax

```
VOID NNTPSearch()
```

Remarks

This search runs as a query against the server and displays in a separate query window.

NNTPThreadChild()

Navigates the current selection to the first child of the currently selected item.

Token ID

```
BFTKN_CHILD_NNTP_THREAD or 1047
```

Syntax

```
VOID NNTPThreadChild()
```

NNTPThreadCollapse()

Collapses the currently selected thread.

Token ID

`BFTKN_COLLAPSE_NNTP_THREAD or 1049`

Syntax

```
VOID NNTPThreadCollapse()
```

NNTPThreadDelete()

Deletes the currently selected thread in a newsgroup.

Token ID

```
BFTKN_DELETE_NNTP_THREAD or 1061
```

Syntax

```
VOID NNTPThreadDelete()
```

NNTPThreadExpand()

Expands the currently selected item.

Token ID

```
BFTKN_EXPAND_NNTP_THREAD or 1048
```

Syntax

```
VOID NNTPThreadExpand()
```

NNTPThreadIgnore()

Marks the currently selected thread "ignore."

Token ID

```
BFTKN_IGNORE_NNTP_THREAD or 1056
```

Syntax

```
VOID NNTPThreadIgnore()
```

Remarks

The thread appears "grayed out" in the display. Only one message in the thread needs to be selected.

NNTPThreadMarkRead()

Marks all messages in the currently selected thread "Read."

Token ID

```
BFTKN_MARK_NNTP_THREAD_READ or 1057
```

Syntax

```
VOID NNTPThreadMarkRead()
```

Remarks

Only one message in the thread needs to be selected.

NNTPThreadParent()

Navigates the current selection to the parent of the currently selected item.

Token ID

```
BFTKN_PARENT_NNTP_THREAD or 1046
```

Syntax

```
VOID NNTPThreadParent()
```

NNTPThreadWatch()

Marks the currently selected thread "watch."

Token ID

```
BFTKN_WATCH_NNTP_THREAD or 1055
```

Syntax

```
VOID NNTPThreadWatch()
```

Remarks

Only one message in the thread needs to be selected.

O

This section contains information about the following tokens:

ODMAQueryDlg()

Displays the ODMA Query dialog box.

Token ID

```
DTKN_QUERY_ODMA or 123
```

Syntax

```
VOID ODMAQueryDlg()
```

ODMSelectDlg()

Displays the ODMA Select dialog box.

Token ID

```
AFTKN_ODM_SELECT_DLG or 871
```

Syntax

```
ANSISTRING ODMSelectDlg([ENUM Select])
```

Parameters

Select As ENUM

(Optional) Enumerated values:

0 No
1 Yes

Return Values

ANSISTRING. DocIDStr ANSISTRING in form Domain.PostOffice.LibraryName:Doc#.Ver# if OK is pressed; nothing if Cancel is pressed. ***APPSELECT*** if "Use Application Dialog" button is pressed.

OfficeCloseViews()

Closes all GroupWise windows except the active window.

Token ID

```
BFTKN_ALLWIN_CLOSE or 200
```

Syntax

```
VOID OfficeCloseViews()
```

OfficeMinimize()

Minimizes all GroupWise windows and views to an icon.

Token ID

```
BFTKN_ALLWIN_MINIMIZE or 201
```

Syntax

```
VOID OfficeMinimize()
```

OleAttachDlg()

Displays the Attach Object dialog box.

Token ID

```
DTKN_ATTACH_OBJECT or 85
```

Syntax

```
VOID OleAttachDlg()
```

OleAttachPaste()

Attaches the contents of the Clipboard to a new item as an OLE object.

Token ID

```
BFTKN_ATTACH_PASTE or 321
```

Syntax

```
VOID OleAttachPaste()
```

OleAttachPasteLink()

Displays dialog allowing a user to select an Ole Object type to attach to the active item compose view.

Token ID

```
BFTKN_ATTACH_PASTELINK or 322
```

Syntax

```
VOID OleAttachPasteLink()
```

OleConvertToStatic()

Converts an OLE embedded object so that it cannot be edited by the recipient.

Token ID

```
BFTKN_CONVERT_TOSTATIC or 294
```

Syntax

```
VOID OleConvertToStatic()
```

OleDoVerb()

Performs an action on an attached or embedded OLE object. The action depends on the object.

Token ID

```
AFTKN_DOVERB or 620
```

Syntax

```
VOID OleDoVerb(ANSISTRING Verb;
               WORD AttachmentIndex)
```

Parameters

Verb As ANSISTRING

Action to perform on an OLE object. If you open an item with two attached objects—a Paintbrush Picture and WordPerfect 6.0 document—the following example opens the second attached object (WP object) in WordPerfect for editing: Application (A1; "WPOffice"; Default; "US") OleDoVerb (Verb: "Edit"; AttachmentIndex: 1)

AttachmentIndex As WORD

Attachment index number (zero-based).

See Also

“OleInsertObjectDlg()” on page 614

OleInsertObject()

Opens an application associated with an OLE object in a custom view.

Token ID

```
AFTKN_INSERT_OBJECT or 619
```

Syntax

```
VOID OleInsertObject( ANSISTRING ObjectClass)
```

Parameters

ObjectClass As ANSISTRING

Object to embed, such as a Paintbrush object.

Remarks

To embed the object in an OLE box, choose File, Update or Update GroupWise depending on the application. The recipient can open, view, edit and/or save an embedded OLE object.

OleInsertObjectDlg()

Displays the insert object dialog box.

Token ID

```
DTKN_INSERT_OBJECT or 73
```

Syntax

```
VOID OleInsertObjectDlg()
```

OleLinksDlg()

Displays the links dialog box to update, cancel, or change a Link.

Token ID

```
DTKN_LINKS or 64
```

Syntax

```
VOID OleLinksDlg()
```

OlePasteLink()

Pastes the contents of the Clipboard into an OLE box, and creates a link to a server application. The object must have been saved in the server application and copied to the Clipboard. You cannot cut the object to the clipboard or make any changes after it is saved and use this command.

Token ID

```
BFTKN_PASTELINK or 210
```

Syntax

```
VOID OlePasteLink()
```

OpenCalendar()

Opens or switches to the default Calendar view.

Token ID

```
BFTKN_OPEN_CALENDAR or 206
```

Syntax

```
VOID OpenCalendar()
```

OpenDocument()

Opens the document identified by the Document Number, specified in DocIDStr. The Value of OpenFlag determines the action OpenDocument will take.

Token ID

```
AFTKN_DM_OPEN or 854
```

Syntax

```
ANSISTRING OpenDocument( ANSISTRING DocIDStr;
                         [ENUM OpenFlag])
```

Parameters

DocIDStr As ANSISTRING

String in the form: Domain.PostOffice.Library:Doc#.Ver.#

OpenFlag As ENUM

(Optional) Specifies how to open the document as follows:

199	LaunchApp
202	UseCurrentApp
203	Quickviewer
204	ReadOnly

Return Values

ANSISTRING. Path and file name of document.

OpenDocumentReadOnly()

Opens the document identified by the Document Number, specified in DocIDStr, in read-only mode. The Value of OpenFlag determines the action OpenDocumentReadOnly will take.

Token ID

```
AFTKN_DM_READONLYOPEN or 857
```

Syntax

```
ANSISTRING OpenDocumentReadOnly( ANSISTRING DocIDStr;
                                 [ENUM OpenFlag])
```

Parameters

DocIDStr As ANSISTRING

String in the form Domain.PostOffice.Library:Doc#.Ver#:

OpenFlag As ENUM

(Optional) Specifies how to open the document as follows:

199	LaunchApp
202	UseCurrentApp
203	Quickviewer
204	ReadOnly

Return Values

ANSISTRING. Path and file name of document.

OpenInBox()

Opens or switches to the Mailbox. Also switches selection and focus to the Mailbox folder.

Token ID

```
BFTKN_OPEN_INBOX or 189
```

Syntax

```
VOID OpenInBox()
```

OpenNewBrowser()

Opens a new main GroupWise client window.

Token ID

```
BFTKN_NEW_BROWSER or 403
```

Syntax

```
VOID OpenNewBrowser()
```

OpenOutBox()

Opens or switches to the Sent Items. Also switches selection and focus to the Mailbox folder. And shows all sent items in that folder.

Token ID

```
BFTKN_OPEN_OUTBOX or 190
```

Syntax

```
VOID OpenOutBox()
```

OpenTrashWindow()

Opens or switches to the Trash window. Also switches selection and focus to the Trash folder. And show all items in that folder.

Token ID

```
BFTKN_OPEN_TRASH or 184
```

Syntax

```
VOID OpenTrashWindow()
```

P

This section contains information about the following tokens:

PasteSpecialView()

Selects options for pasting from the Clipboard.

Token ID

```
BFTKN_VIEW_PASTESPECIAL or 263
```

Syntax

```
VOID PasteSpecialView()
```

PosCharNext()

Moves the insertion point to the next character or space.

Token ID

```
BFTKN_RIGHTARROW or 230
```

Syntax

```
VOID PosCharNext()
```

Remarks

The message body field (in the Message View and the Object API) has a limit of 32,000 characters. The Message View will stop accepting data after 32,000 characters.

PosCharPrevious()

Moves the insertion point to the previous character or space.

Token ID

```
BFTKN_LEFTARROW or 223
```

Syntax

```
VOID PosCharPrevious()
```

Remarks

The message body field (in the Message View and the Object API) has a limit of 32,000 characters. The Message View will stop accepting data after 32,000 characters.

PosLineBegin()

Moves the insertion point to the beginning of the current line.

Token ID

```
BFTKN_BEGLINE or 212
```

Syntax

```
VOID PosLineBegin()
```

Remarks

The message body field (in the Message View and the Object API) has a limit of 32,000 characters. The Message View will stop accepting data after 32,000 characters.

PosLineDown()

Moves the insertion point down one line.

Token ID

```
BFTKN_DOWNarrow or 220
```

Syntax

```
VOID PosLineDown()
```

Remarks

The message body field (in the Message View and the Object API) has a limit of 32,000 characters. The Message View will stop accepting data after 32,000 characters.

PosLineEnd()

Moves the insertion point to the end of the current line.

Token ID

```
BFTKN_ENDLINE or 221
```

Syntax

```
VOID PosLineEnd()
```

Remarks

The message body field (in the Message View and the Object API) has a limit of 32,000 characters. The Message View will stop accepting data after 32,000 characters.

PosLineUp()

Moves the insertion point up one line.

Token ID

```
BFTKN_UPARROW or 245
```

Syntax

```
VOID PosLineUp()
```

Remarks

The message body field (in the Message View and the Object API) has a limit of 32,000 characters. The Message View will stop accepting data after 32,000 characters.

PosScreenDown()

Displays the next screen. The insertion point stays on the same line and position, if possible.

Token ID

```
BFTKN_PAGEDOWN or 224
```

Syntax

```
VOID PosScreenDown()
```

Remarks

The message body field (in the Message View and the Object API) has a limit of 32,000 characters. The Message View will stop accepting data after 32,000 characters.

PosScreenLeft()

Moves the insertion point to the left side of a message box.

Token ID

```
BFTKN_PAGELEFT or 225
```

Syntax

```
VOID PosScreenLeft()
```

Remarks

The message body field (in the Message View and the Object API) has a limit of 32,000 characters. The Message View will stop accepting data after 32,000 characters.

PosScreenRight()

Moves the insertion point to the end of the current line.

Token ID

```
BFTKN_PAGERIGHT or 226
```

Syntax

```
VOID PosScreenRight()
```

Remarks

The message body field (in the Message View and the Object API) has a limit of 32,000 characters. The Message View will stop accepting data after 32,000 characters.

PosScreenUp()

Displays the previous screen. The insertion point stays on the same line and position, if possible.

Token ID

```
BFTKN_PAGEUP or 227
```

Syntax

```
VOID PosScreenUp()
```

Remarks

The message body field (in the Message View and the Object API) has a limit of 32,000 characters. The Message View will stop accepting data after 32,000 characters.

PosTextTop()

Moves the insertion point to the beginning of the first line.

Token ID

```
BFTKN_BEGTEXT or 213
```

Syntax

```
VOID PosTextTop()
```

Remarks

The message body field (in the Message View and the Object API) has a limit of 32,000 characters. The Message View will stop accepting data after 32,000 characters.

PosToEndOfText()

Moves the insertion point to the end of the last line.

Token ID

```
BFTKN_ENDTEXT or 222
```

Syntax

```
VOID PosToEndOfText()
```

Remarks

The message body field (in the Message View and the Object API) has a limit of 32,000 characters. The Message View will stop accepting data after 32,000 characters.

PosWordLeft()

Moves the insertion point to the beginning of the current word, or to the beginning of the previous word, depending on its current position.

Token ID

```
BFTKN_WORDLEFT or 246
```

Syntax

```
VOID PosWordLeft()
```

Remarks

The message body field (in the Message View and the Object API) has a limit of 32,000 characters. The Message View will stop accepting data after 32,000 characters.

PosWordRight()

Moves the insertion point to the beginning of the next word.

Token ID

```
BFTKN_WORDRIGHT or 247
```

Syntax

```
VOID PosWordRight()
```

Remarks

The message body field (in the Message View and the Object API) has a limit of 32,000 characters. The Message View will stop accepting data after 32,000 characters.

PrefAdvanced()

Sets the default Advanced Send options.

Token ID

```
AFTKN_SET_ADV or 594
```

Syntax

```
VOID PrefAdvanced( [ENUM Security];
                   [ENUM ConcealSubject];
                   [WORD DeliverDelay];
                   [ENUM InsertOutbox];
                   [ENUM ConvertAttachments];
                   [ENUM RequireRoutedPassword];
                   [WORD DeliverHour];
                   [WORD DeliverMinute];
                   [WORD DeliverDay];
                   [WORD DeliverMonth];
                   [WORD DeliverYear];
                   [ENUM InternetDeliveryStatus];
                   [ENUM InternetReceiptMailbox];
                   [ENUM SignatureRequired];
                   [ENUM EncryptMessage];
                   [ENUM MimeEncoding])
```

Parameters

Security As ENUM

(Optional) Security label appears on the first line of a message box:

1	Normal
2	Proprietary
3	Confidential
4	Secret
5	TopSecret
6	ForYourEyesOnly

ConcealSubject As ENUM

(Optional) Conceal Mailbox and Sent Items subject lines:

0 No
1 Yes

DeliverDelay As WORD

(Optional) Number of days to delay delivery.

InsertOutbox As ENUM

(Optional) Insert items in sender's Sent Items:

0 No
1 Yes

ConvertAttachments As ENUM

(Optional) Convert attached files through a gateway:

0 No
1 Yes

RequireRoutedPassword As ENUM

(Optional) Required password to complete routed items:

0 No
1 Yes

DeliverHour As WORD

(Optional) Hour to deliver a delay delivery item.

DeliverMinute As WORD

(Optional) Minute to deliver a delay delivery item.

DeliverDay As WORD

(Optional) Day to deliver a delay delivery item.

DeliverMonth As WORD

(Optional) Month to deliver a delay delivery item.

DeliverYear As WORD

(Optional) Year to deliver a delay delivery item.

InternetDeliveryStatus As ENUM

(Optional) Enumerated values:

0 No
1 Yes

InternetReceiptMailbox As ENUM

(Optional) Enumerated values:

0 No
1 Yes

SignatureRequired As ENUM

(Optional) Enumerated values:

0 No
1 Yes

EncryptMessage As ENUM

(Optional) Enumerated values:

0 No
1 Yes

MimeEncoding As ENUM

(Optional) Enumerated values:

1 Windows
2 ISO
3 UTF8
4 ARABIC_ASMO_708

PrefAppearance()

Sets the client's appearance.

Token ID

```
AFTKN_SET_APPEARANCE or 925
```

Syntax

```
VOID PrefAppearance([ENUM ShowMainMenu];
                    [ENUM ShowNavbar];
                    [ENUM ShowMainToolbar];
                    [ENUM ShowFolderList];
                    [ENUM FolderListViewType];
                    [ENUM FolderListType];
                    [ENUM ShowQV];
                    [ENUM QVPosition];
                    [ENUM UseColorSchemes];
                    [ENUM ColorScheme])
```

Parameters

ShowMainMenu As ENUM

(Optional)

0 No
1 Yes

ShowNavbar As ENUM

(Optional)

0 No
1 Yes

ShowMainToolbar As ENUM

(Optional)

0 No
1 Yes

ShowFolderList As ENUM

(Optional)

0 No
1 Yes

FolderListViewType As ENUM

(Optional)

0 Simple
1 Full

FolderListViewType As ENUM

(Optional)

0 Simple
1 Full

FolderListType As ENUM

(Optional)

0 Short
1 Long

ShowQV As ENUM

(Optional)

0 No
1 Yes

QVPosition As ENUM

(Optional)

0 Bottom
1 Right

UseColorSchemes As ENUM

(Optional)

0 No
1 Yes

ColorScheme As ENUM

(Optional)

0 Blue
1 OliveGreen
2 Silver
3 SkyBlue
4 SpringGreen
5 SterlingSilver

PrefAppointment()

Sets the default Appointment Send options.

Token ID

```
AFTKN_SET_APPT or 597
```

Syntax

```
VOID PrefAppointment( [ENUM Priority];
                      [ENUM Notify];
                      [ENUM ReturnOnOpen];
                      [ENUM ReturnOnAccept];
                      [ENUM ReturnOnDelete];
                      [ENUM StatusInfo])
```

Parameters

Priority As ENUM

(Optional) Specifies the priority:

0	Low
1	Normal
2	High

Notify As ENUM

Notify a recipient when an Appointment item is delivered to his Mailbox (optional):

0 No
1 Yes

ReturnOnOpen As ENUM

Return notification options (optional):

276	Mail
3	MailAndNotify
0	None
2	Notify
1	zOldMail

ReturnOnAccept As ENUM

Return notification options for accepted or completed items (optional):

276	Mail
3	MailAndNotify
0	None
2	Notify
1	zOldMail

ReturnOnDelete As ENUM

Return notification options (optional):

276	Mail
3	MailAndNotify
0	None
2	Notify
1	zOldMail

StatusInfo As ENUM

Information to track when an Appointment item is sent (optional):

3	AllInfo
275	Delivered
0	None
280	Opened

See Also

"PrefDlg()" on page 654

PrefAppointmentTime()

Sets the default Appointment Time options.

Token ID

```
AFTKN_SET_VEW or 608
```

Syntax

```
VOID PrefAppointmentTime( [ENUM ShowTimeIntervals];
                      [WORD StartTimeMinute];
                      [WORD StartTimeHour];
                      [WORD Interval];
                      [ENUM Alarm];
                      [WORD AlarmMinutes];
                      [WORD AlarmHours];
                      [ENUM DisplayEventType];
                      [WORD DefaultLengthMinutes];
                      [WORD DefaultLengthHours];
                      [ENUM IncludeSelf])
```

Parameters

ShowTimeIntervals As ENUM

(Optional) Enumerated values:

0 No
1 Yes

StartTimeMinute As WORD

(Optional) Minute to begin day on.

StartTimeHour As WORD

(Optional) Hour to begin day on.

Interval As WORD

(Optional) Length of display time interval.

Alarm As ENUM

(Optional) Automatically turns on alarm when an appointment is accepted:

0 No
1 Yes

AlarmMinutes As WORD

(Optional) Minutes before appointment to sound alarm.

AlarmHours As WORD

(Optional) Hours before appointment to sound alarm.

DisplayEventType As ENUM

(Optional) Specifies how to display the appointment length:

0 Duration
1 EndDateAndTime

DefaultLengthMinutes As WORD

(Optional) Default length of appointment in minutes.

DefaultLengthHours As WORD

(Optional) Default length of appointment in hours.

IncludeSelf As ENUM

(Optional) Enumerated values:

0 No
1 Yes

See Also

“PrefDlg()” on page 654

PrefBusySearch()

Sets the default Busy Search options.

Token ID

```
AFTKN_SET_SCH or 615
```

Syntax

```
VOID PrefBusySearch( [WORD StartTimeMinute];
                     [WORD StartTimeHour];
                     [WORD EndTimeMinute];
                     [WORD EndTimeHour];
                     [WORD AppointmentLengthMinutes];
                     [WORD AppointmentLengthHours];
                     [ENUM BusySunday];
                     [ENUM BusyMonday];
                     [ENUM BusyTuesday];
                     [ENUM BusyWednesday];
                     [ENUM BusyThursday];
                     [ENUM BusyFriday];
                     [ENUM BusySaturday];
                     [ENUM ExtendedInformation];
                     [WORD SearchRange])
```

Parameters

StartTimeMinute As WORD

(Optional) Minute to start Busy Search display on.

StartTimeHour As WORD

(Optional) Hour to start Busy Search display on.

EndTimeMinute As WORD

(Optional) Minute to end Busy Search display on.

EndTimeHour As WORD

(Optional) Hour to end Busy Search display on.

AppointmentLengthMinutes As WORD

(Optional) Number of minutes in the appointment length.

AppointmentLengthHours As WORD

(Optional) Number of hours in the appointment length.

BusySunday As ENUM

Include Sunday in the search (optional):

0 No
1 Yes

BusyMonday As ENUM

Include Monday in the search (optional):

0 No
1 Yes

BusyTuesday As ENUM

Include Tuesday in the search (optional):

0 No
1 Yes

BusyWednesdayAs ENUM

Include Wednesday in the search (optional):

0 No
1 Yes

BusyThursday As ENUM

Include Thursday in the search (optional):

0 No
1 Yes

BusyFriday As ENUM

Include Friday in the search (optional):

0 No
1 Yes

BusySaturday As ENUM

Include Saturday in the search (optional):

0 No
1 Yes

ExtendedInformation As ENUM

Display additional appointment information, depending on your access rights (optional):

0 No
1 Yes

SearchRange As WORD

Number of days to search (optional).

See Also

"PrefDlg()" on page 654

PrefCleanup()

Sets the default Cleanup options.

Token ID

```
AFTKN_SET_DSC or 631
```

Syntax

```
VOID PrefCleanup( [ENUM MailAndPhoneDelete];
                  [WORD MailAndPhoneDelayDays];
                  [ENUM AppointmentTaskAndNoteDelete];
                  [WORD AppointmentTaskAndNoteDelayDays];
                  [ENUM EmptyTrash];
                  [WORD EmptyTrashDelayDays])
```

Parameters

MailAndPhoneDelete As ENUM

(Optional) Specifies the deletion option as follows:

2	AutoArchive
1	AutoDelete
0	Manual

MailAndPhoneDelayDays As WORD

(Optional) Number of days to AutoArchive! or AutoDelete! cleanup.

AppointmentTaskAndNoteDelete As ENUM

(Optional) Specifies the deletion option as follows:

2	AutoArchive
1	AutoDelete
0	Manual

AppointmentTaskAndNoteDelayDays As WORD

Number of days to AutoArchive! or AutoDelete! cleanup (optional).

EmptyTrash As ENUM

(Optional)

1 Automatic
0 Manual

EmptyTrashDelayDays As WORD

Number of days to AutoArchive! or AutoDelete! cleanup (optional).

PrefDateTimeFormat()

Sets the default Date and Time format.

Token ID

```
AFTKN_SET_DTM or 613
```

Syntax

```
VOID PrefDateTimeFormat( [ANSISTRING DateDefault];
                         [ANSISTRING TimeDefault];
                         [ANSISTRING MailboxDate];
                         [ANSISTRING InfoDate];
                         [ANSISTRING FileDate])
```

Parameters

DateDefault As ANSISTRING

Date format for all GroupWise views (optional).

TimeDefault As ANSISTRING

Time format for all GroupWise views. (Optional)

MailboxDate As ANSISTRING

Mailbox date and time formats. (Optional)

InfoDate As ANSISTRING

Info date and time formats. To see the date, open Mailbox, choose Actions > Info. (Optional)

FileDate As ANSISTRING

Info date and time formats for attached items. To see the date, open Sent Items, choose Actions > Info. (Optional)

See Also

“PrefDlg()” on page 654

PrefDlg()

Displays the preference dialog box to set GroupWise defaults.

Token ID

```
DTKN_PRESETUP or 60
```

Syntax

```
VOID PrefDlg()
```

PrefEnvironment()

Sets the default Environment options.

Token ID

```
AFTKN_SET_ENV or 611
```

Syntax

```
VOID PrefEnvironment( [ANSISTRING Language];
                      [ENUM AdvanceOnDelete];
                      [ENUM NewScreenOnSend];
                      [WORD UpdateRateSeconds];
                      [WORD UpdateRateMinutes];
                      [ENUM OpenItemAction];
                      [ENUM OpenAttachmentAction];
                      [ENUM Mnemonics];
                      [ENUM PromptEmptyFilter];
                      [ENUM PromptEmptyFind];
                      [ENUM AutoSpellCheck];
                      [ENUM LaunchNotify])
```

Parameters

Language As ANSISTRING

(Optional) The display language that affects only the interface. The desired language version must be installed. After setting, exit and re-enter GroupWise.

AdvanceOnDelete As ENUM

(Optional) Display the next Mailbox or Sent Items item after you accept, decline, or delete the current item:

0 No
1 Yes

NewScreenOnSend As ENUM

(Optional) Open an new item view after item is sent:

0 No
1 Yes

UpdateRateSeconds As WORD

(Optional) Specify in seconds how frequently GroupWise checks your mailbox for incoming items.

UpdateRateMinutes As WORD

(Optional) Specify in minutes how frequently GroupWise checks your mailbox for incoming items.

OpenItemAction As ENUM

(Optional) Specifies the action:

1	OpenItem
2	ResendItem
0	ShowProperties

OpenAttachmentAction As ENUM

(Optional) Enumerated values:

1 OpenAttachment
0 ViewAttachment

Mnemonics As ENUM

(Optional) Enumerated values:

0 No
1 Yes

PromptEmptyFilter As ENUM

(Optional) Enumerated values:

0 No
1 Yes

PromptEmptyFind As ENUM

(Optional) Enumerated values:

0 No
1 Yes

AutoSpellCheck As ENUM

(Optional) Enumerated values:

0 No
1 Yes

LaunchNotify As ENUM

(Optional) Enumerated values:

0 No
1 Yes

PrefLocationOfFiles()

Specifies preferences for file locations.

Token ID

```
AFTKN_SET_LOC or 616
```

Syntax

```
VOID PrefLocationOfFiles( [ANSISTRING ArchiveDir];
                          [ANSISTRING CustomViews];
                          [ANSISTRING DefaultSaveDir];
                          [ANSISTRING BackupDir];
                          [WORD BackupDaysBetween];
                          [ANSISTRING CachingDir];
                          [ENUM CopyArchive])
```

Parameters

ArchiveDir As ANSISTRING

(Optional) Location of the archive directory.

CustomViews As ANSISTRING

(Optional) Location of the custom views directory.

DefaultSaveDir As ANSISTRING

(Optional) Location for item files and attachments.

BackupDir As ANSISTRING

(Optional) Location for backups.

BackupDaysBetween As WORD

(Optional) Backup interval.

CachingDir As ANSISTRING

(Optional) Location for Caching directory.

CopyArchive As ENUM

(Optional) Enumerated values:

Number	Value	Description
0	NO	Change the archive location without copying the archives.
1	YES	Change the archive location and copy the archives to the new location.
201	PROMPT	(Default) Prompt the user to choose whether they want the archives copied.

See Also

"PrefDlg()" on page 654

PrefMailAndPhone()

Sets the default Mail and Phone Send options.

Token ID

```
AFTKN_SET_MESSAGE or 596
```

Syntax

```
VOID PrefMailAndPhone( [ENUM Priority];
                       [ENUM Notify];
                       [ENUM ReturnOnOpen];
                       [ENUM ReturnOnCompleted];
                       [ENUM ReturnOnDelete];
                       [ENUM StatusInfo];
                       [ENUM ReplyRequested];
                       [WORD ReplyRequestedDays];
                       [WORD ExpireDays];
                       [ENUM AutoDelete])
```

Parameters

Priority As ENUM

(Optional) Specifies the priority:

0	Low
1	Normal
2	High

Notify As ENUM

Notify recipients when a mail or phone item is delivered to their Mailbox. (Optional)

0 No
1 Yes

ReturnOnOpen As ENUM

Return notification options for routed items. (Optional)

276	Mail
3	MailAndNotify
0	None
2	Notify
1	zOldMail1

ReturnOnCompleted As ENUM

Return notification options. (Optional)

276	Mail
3	MailAndNotify
0	None
2	Notify
1	zOldMail1

ReturnOnDelete As ENUM

Return notification options. (Optional)

276	Mail
3	MailAndNotify
0	None
2	Notify
1	zOldMail1

StatusInfo As ENUM

Information to track when an Appointment item is sent. (Optional)

3	AllInfo
275	Delivered
0	None
280	Opened

ReplyRequested As ENUM

Request reply on the first line of the message box. (Optional)

0	None
1	WhenConvenient
2	WithinDays

ReplyRequestedDays As WORD

Number of days. The recipient sees: REPLY REQUESTED: By xx/xx/xx. (Optional)

ExpireDays As WORD

Number of days a mail or phone message remains in recipient's mailbox. (Optional)

AutoDelete As ENUM

Delete the item from the sender’s Sent Items when deleted from the recipient’s Mailbox. (Optional)

0 No
1 Yes

See Also

“PrefDlg()” on page 654

PrefNote()

Sets the default Note Send options.

Token ID

```
AFTKN_SET_NOTE or 598
```

Syntax

```
VOID PrefNote( [ENUM Priority];
               [ENUM Notify];
               [ENUM ReturnOnOpen];
               [ENUM ReturnOnDelete];
               [ENUM StatusInfo])
```

Parameters

Priority As ENUM

(Optional) Specifies the priority:

0	Low
1	Normal
2	High

Notify As ENUM

Notify recipients when a Note is delivered to their Mailbox. (Optional)

0 No
1 Yes

ReturnOnOpen As ENUM

Return notification options for routed items. (Optional)

276	Mail
3	MailAndNotify
0	None
2	Notify
1	zOldMaill

ReturnOnDelete As ENUM

Return notification options. (Optional)

276	Mail
3	MailAndNotify
0	None
2	Notify
1	zOldMailI

StatusInfo As ENUM

Information to track when an Appointment is sent. (Optional)

3	AllInfo
275	Delivered
0	None
280	Opened

See Also

"PrefDlg()" on page 654

PrefSignature()

Sets the signature that is appended to every sent message.

Token ID

```
AFTKN_SET_SIG or 885
```

Syntax

```
VOID PrefSignature( [ENUM SignatureOptions];
                    [ENUM AddSignature];
                    [ENUM AddVCard];
                    [ANSISTRING SignatureText];
                    [ANSISTRING VCardFilename];
                    [ANSISTRING AccountName];
                    [ANSISTRING AccountRecordId])
```

Parameters

SignatureOptions As ENUM

(Optional) Specifies the mode for adding signatures or vCard* information:

0 Do not add
1 Automatically add
201 Prompt before adding

AddSignature As ENUM

(Optional) Specifies whether or not to add the signature:

0 No
1 Yes

AddVCard As ENUM

(Optional) Specifies whether or not to add a vCard:

0 No
1 Yes

SignatureText As ANSISTRING

(Optional) Specifies the signature text.

VCardFilename As ANSISTRING

(Optional) Specifies the name of the vCard file. If the name isn't provided, the vCard information is generated by the Address Book.

AccountName As ANSISTRING

(Optional) Specifies the name of the sending account. For example, GroupWise.

AccountRecordId As ANSISTRING

(Optional) Specifies the record ID of the sending account.

Remarks

You cannot turn off the signature option by setting only SignatureOptions to 0. If you want to turn off signatures, you must also set AddSignature to 0, as shown in the following example:

```
PrefSignature(0;0);
```

See Also

“PrefDlg()” on page 654

PrefTask()

Sets the default Task Send options.

Token ID

```
AFTKN_SET_TODO or 599
```

Syntax

```
VOID PrefTask( [ENUM Priority];
               [ENUM Notify];
               [ENUM ReturnOnOpen];
               [ENUM ReturnOnAccept];
               [ENUM ReturnOnDelete];
               [ENUM ReturnOnCompleted];
               [ENUM StatusInfo];
               [ENUM DisplayEventType])
```

Parameters

Priority As ENUM

(Optional) Specifies the priority:

0	Low
1	Normal
2	High

Notify As ENUM

Notify recipients when a Task item is delivered to their Mailbox. (Optional)

0 No
1 Yes

ReturnOnOpen As ENUM

Return notification options for routed items. (Optional)

ReturnOnAccept As ENUM

Return notification options. (Optional)

276	Mail
3	MailAndNotify
0	None
2	Notify
1	zOldMaill

ReturnOnDelete As ENUM

Return notification options. (Optional)

276	Mail
3	MailAndNotify
0	None
2	Notify
1	zOldMaill

ReturnOnCompleted As ENUM

Return notification options. (Optional)

276	Mail
3	MailAndNotify
0	None
2	Notify
1	zOldMaill

StatusInfo As ENUM

Information to track when an Appointment item is sent. (Optional)

3	AllInfo
275	Delivered
0	None
280	Opened

DisplayEventType As ENUM

Specify how to display a task due date. (Optional)

0 Duration
1 EndDateAndTime

See Also

"PrefDlg()" on page 654

PrefViewDefaults()

Sets the default views for personal and group views.

Token ID

```
AFTKN_SET_DEF_VIEW or 633
```

Syntax

```
PrefViewDefaults( [ANSISTRING GroupAppointmentView];
                  [ANSISTRING PersonalAppointmentView];
                  [ANSISTRING FolderView];
                  [ANSISTRING MailView];
                  [ANSISTRING GroupNoteView];
                  [ANSISTRING PersonalNoteView];
                  [ANSISTRING PhoneView];
                  [ANSISTRING GroupTaskView];
                  [ANSISTRING PersonalTaskView];
                  [ANSISTRING PostedMailView];
                  [ANSISTRING PostedPhoneView];
                  [ENUM ReadUsingDefaultViews])
```

Parameters

GroupAppointmentView As ANSISTRING

Example: "Meeting w/attach" (Optional)

PersonalAppointmentView As ANSISTRING

Example: "Personal Appointment w/attach" (Optional)

FolderView As ANSISTRING

Calendar View. Example: "Day Planner" (Optional)

MailView As ANSISTRING

Example: "Expanded Mail" (Optional)

GroupNoteView As ANSISTRING

Example: "Notice" (Optional)

PersonalNoteView As ANSISTRING

Example: "Personal Note" (Optional)

PhoneView As ANSISTRING

Example: "Phone w/attach" (Optional)

GroupTaskView As ANSISTRING

Example: "Task" (Optional)

PersonalTaskView As ANSISTRING

Example: "Personal Task" (Optional)

PostedMailView As ANSISTRING

Example: “Posted Mail” (Optional)

PostedPhoneView As ANSISTRING

Example: “Posted Phone Message” (Optional)

ReadUsingDefaultViews As ENUM

(Optional)

0 No
1 Yes

PreviousThread()

Moves the cursor to the previous thread.

Token ID

```
BFTKN_PREV_THREAD or 389
```

Syntax

```
VOID PreviousThread()
```

PreviousUnreadItem()

Move the cursor to the previous unread item.

Token ID

```
BFTKN_PREV_UNREAD or 391
```

Syntax

```
VOID PreviousUnreadItem()
```

Remarks

Moves the cursor to the previous unread item.

Print()

Prints an opened attachment.

Token ID

```
BFTKN_PRINT or 165
```

Syntax

```
VOID Print()
```

PrintCalendarDlg()

Displays the Print Calendar dialog box.

Token ID

```
DTKN_PRINT_CALENDAR or 43
```

Syntax

```
VOID PrintCalendarDlg()
```

PrintContactDlg()

Displays a dialog enabling the user to print the currently displayed list of contacts.

Token ID

```
DTKN_PRINT_CONTACTS or 155
```

Syntax

```
VOID PrintContactDlg()
```

PrintDlg()

Displays the Print dialog box.

Token ID

```
DTKN_PRINT or 83
```

Syntax

```
VOID PrintDlg()
```

PrintListDlg()

Displays a dialog enabling the user to print the currently displayed list of items.

Token ID

```
DTKN_PRINT_LIST or 153
```

Syntax

```
VOID PrintListDlg()
```

PrintSetup()

Displays the Print Setup dialog box.

Token ID

```
DTKN_PRINT_SETUP or 55
```

Syntax

```
VOID PrintSetup()
```

PromptForPassword()

Prompts the user for a password and verifies that it is correct. User may enter a password as many as three times.

Token ID

```
AFTKN_PROMPT_FOR_PASSWORD or 711
```

Syntax

```
VOID PromptForPassword()
```

PromptSetMode()

Specifies whether to suppress macro prompt messages.

Token ID

```
BFTKN_SET_PROMPT or 374
```

Syntax

```
VOID PromptSetMode( ENUM PromptingMode)
```

Parameters

PromptingMode As ENUM

1 Normal
226 Suppressed

Remarks

Prompt messages that cannot be anticipated or responded to can interfere with DDE macro playback. When PromptingMode is Suppressed!, macros perform default actions and do not display a prompt message.

This command which does not affect error messages, is automatically set to Normal! when the macro or DDE conversation ends.

Properties()

Invokes the Folder Properties dialog for the currently selected folder, or the folder specified with the "FolderName" parameter. If an item is selected, it displays the properties dialog for that item. For a new item, this token invokes the send options dialog.

Token ID

```
DTKN_PROPERTIES or 68
```

Syntax

```
VOID Properties([ANSISTRING FolderName])
```

Parameters

FolderName As ANSISTRING

(Optional) Provide a fully qualified folder name.

Proxy()

Specifies a person to proxy for (the person must grant proxy rights).

Token ID

```
AFTKN_CHANGE_USER or 624
```

Syntax

```
VOID Proxy( ANSISTRING UserID;
            [ENUM Window];
            [ANSISTRING UserGUID])
```

Parameters

UserID As ANSISTRING

Person to proxy for.

Window As ENUM

(Optional) Enumerated values:

272 Current Window

UserGUID As ANSISTRING

(Optional)

ProxyDlg()

Displays the Proxy dialog box.

Token ID

```
DTKN_PROXY or 80
```

Syntax

```
VOID ProxyDlg()
```

Q-R

This section contains information about the following tokens:

QueryDlg()

Displays the QuickFinder dialog box.

Token ID

```
DTKN_QUERY or 97
```

Syntax

```
VOID QueryDlg()
```

QueryEditDlg()

Displays the Find Criteria dialog box. The Find Results window must be showing for this token to work.

Token ID

```
DTKN_QUERY_EDIT or 100
```

Syntax

```
VOID QueryEditDlg()
```

QueryExecute()

Executes a named query.

Token ID

`AFTKN_QUERY_EXECUTE or 849`

Syntax

```
VOID QueryExecute( [ANSISTRING QuickFinderQuery];
                   [DWORD Filter];
                   [ENUM WhereToSearch];
                   [ENUM OfficialVersion];
                   [ANSISTRING Libraries];
                   [ANSISTRING FullUserID];
                   [DWORD FolderListHandle];
                   [ANSISTRING FullUserID];
                   [DWORD FolderListHandle];
                   [ANSISTRING FullUserID];
                   [DWORD FolderListHandle];
                   [ANSISTRING FullUserID];
                   [DWORD FolderListHandle];
                   [ANSISTRING FullUserID];
                   [DWORD FolderListHandle];
                   [ANSISTRING FullUserID];
                   [DWORD FolderListHandle];
                   [ANSISTRING FullUserID];
                   [DWORD FolderListHandle];
                   [ANSISTRING FullUserID];
                   [DWORD FolderListHandle];
                   [ANSISTRING FullUserID];
                   [DWORD FolderListHandle];
                   [ANSISTRING FullUserID];
                   [DWORD FolderListHandle];
                   [ANSISTRING FullUserID];
                   [DWORD FolderListHandle];
                   [ANSISTRING FullUserID];
                   [DWORD FolderListHandle];
                   [ANSISTRING FullUserID];
                   [DWORD FolderListHandle];
                   [ANSISTRING FullUserID];
                   [DWORD FolderListHandle];
                   [ANSISTRING FullUserID];
                   [DWORD FolderListHandle];
                   [ANSISTRING FullUserID];
                   [DWORD FolderListHandle];
                   [ANSISTRING FullUserID];
                   [DWORD FolderListHandle];
                   [ANSISTRING FullUserID];
                   [DWORD FolderListHandle];
                   [ANSISTRING FullUserID];
                   [DWORD FolderListHandle];
                   [ANSISTRING FullUserID];
                   [DWORD FolderListHandle])
```

Parameters

QuickFinderQuery As ANSISTRING

(Optional) Name of the query.

Filter As DWORD

(Optional) Filter associated with the query.

WhereToSearch As ENUM

(Optional) Enumerated values:

2 RemoteMailBox
4 MasterMailBox
8 NNTP
16 IMAP

OfficialVersion As ENUM

(Optional) Enumerated values:

654 OfficialOnly

Libraries As ANSISTRING

(Optional). Separates multiple libraries by a comma. In the form: Library.PostOffice.Domain.

FullUserID As ANSISTRING

(Optional) ID of the main user.

FolderListHandle As DWORD

(Optional)

FullUserID As ANSISTRING

(Optional) ID of a proxy user.

FolderListHandle As DWORD

(Optional)

FullUserID As ANSISTRING

(Optional) ID of a proxy user.

FolderListHandle As DWORD

(Optional)

FullUserID As ANSISTRING

(Optional) ID of a proxy user.

FolderListHandle As DWORD

(Optional)

FullUserID As ANSISTRING

(Optional) ID of a proxy user.

FolderListHandle As DWORD

(Optional)

FullUserID As ANSISTRING

(Optional) ID of a proxy user.

FolderListHandle As DWORD

(Optional)

FullUserID As ANSISTRING

(Optional) ID of a proxy user.

FolderListHandle As DWORD

(Optional)

FullUserID As ANSISTRING

(Optional) ID of a proxy user.

FolderListHandle As DWORD

(Optional)

FullUserID As ANSISTRING

(Optional) ID of a proxy user.

FolderListHandle As DWORD

(Optional)

FullUserID As ANSISTRING

(Optional) ID of a proxy user.

FolderListHandle As DWORD

(Optional)

FullUserID As ANSISTRING

(Optional) ID of a proxy user.

FolderListHandle As DWORD

(Optional)

FullUserID As ANSISTRING

(Optional) ID of a proxy user.

FolderListHandle As DWORD

(Optional)

FullUserID As ANSISTRING

(Optional) ID of a proxy user.

FolderListHandle As DWORD

(Optional)

FullUserID As ANSISTRING

(Optional) ID of a proxy user.

FolderListHandle As DWORD

(Optional)

FullUserID As ANSISTRING

(Optional) ID of a proxy user.

FolderListHandle As DWORD

(Optional)

FullUserID As ANSISTRING

(Optional) ID of a proxy user.

FolderListHandle As DWORD

(Optional)

FullUserID As ANSISTRING

(Optional) ID of a proxy user.

FolderListHandle As DWORD

(Optional)

FullUserID As ANSISTRING

(Optional) ID of a proxy user.

FolderListHandle As DWORD

(Optional)

FullUserID As ANSISTRING

(Optional) ID of a proxy user.

FolderListHandle As DWORD

(Optional)

FullUserID As ANSISTRING

(Optional) ID of a proxy user.

FolderListHandle As DWORD

(Optional)

FullUserID As ANSISTRING

(Optional) ID of a proxy user.

FolderListHandle As DWORD

(Optional)

Remarks

The FullUserID and FolderListHandle parameters are tied together. If you desire to use the FolderListHandle parameter you must also specify the FullUserID, otherwise the query will search all folders and the default library.

The first pair of FullUserIds and FolderListHandles are for the main user. Subsequent pairs are for different proxy individuals.

QueryExecuteEx()

Execute a query with additional options for each search source.

Token ID

AFTKN_QUERY_EXECUTEEX or 901

Syntax

```
VOID QueryExecuteEX(
     [ANSISTRING QuickFinderQuery];
     [DWORD Filter];
     [ENUM WhereToSearch];
     [ENUM OfficialVersion];
     [ANSISTRING Libraries];
     [ANSISTRING FullUserID];
     [DWORD FolderListHandle];
     [DWORD SearchOption];
     [ANSISTRING FullUserID];
     [DWORD FolderListHandle];
     [DWORD SearchOption];
     [ANSISTRING FullUserID];
     [DWORD FolderListHandle];
     [ANSISTRING FullUserID];
     [DWORD SearchOption];
     [DWORD FolderListHandle];
     [ANSISTRING FullUserID];
     [DWORD FolderListHandle];
     [DWORD SearchOption];
     [ANSISTRING FullUserID];
     [DWORD FolderListHandle];
     [DWORD SearchOption];
     [ANSISTRING FullUserID];
     [DWORD FolderListHandle];
     [DWORD SearchOption];
     [ANSISTRING FullUserID];
     [DWORD FolderListHandle];
     [DWORD SearchOption];
     [ANSISTRING FullUserID];
     [DWORD FolderListHandle];
     [DWORD SearchOption];
     [ANSISTRING FullUserID];
     [DWORD FolderListHandle];
     [DWORD SearchOption];
     [ANSISTRING FullUserID];
     [DWORD FolderListHandle];
     [DWORD SearchOption];
     [ANSISTRING FullUserID];
     [DWORD FolderListHandle];
     [DWORD SearchOption];
     [ANSISTRING FullUserID];
     [DWORD FolderListHandle];
     [DWORD SearchOption];
     [ANSISTRING FullUserID];
     [DWORD FolderListHandle];
     [DWORD SearchOption];
     [ANSISTRING FullUserID];
```

```
[DWORD FolderListHandle];
[DWORD SearchOption];
[ANSISTRING FullUserID];
[DWORD FolderListHandle];
[DWORD SearchOption];
[ANSISTRING FullUserID];
[DWORD FolderListHandle];
[DWORD SearchOption];
[ANSISTRING FullUserID];
[DWORD FolderListHandle];
[DWORD SearchOption];
[ANSISTRING FullUserID];
[DWORD FolderListHandle];
[DWORD SearchOption];
[ANSISTRING FullUserID];
[DWORD FolderListHandle];
[DWORD SearchOption])
```

Parameters

QuickFinderQuery As ANSISTRING

(Optional) Name of the query.

Filter As DWORD

(Optional) Filter associated with the query.

WhereToSearch As ENUM

(Optional) Enumerated values:

2 RemoteMailBox
4 MasterMailBox
8 NNTP
16 IMAP

OfficialVersion As ENUM

(Optional) Enumerated values:

654 OfficialOnly

Libraries As ANSISTRING

(Optional). Separates multiple libraries by a comma. In the form: Library.PostOffice.Domain.

FullUserID As ANSISTRING

(Optional) ID of the main user.

FolderListHandle As DWORD

(Optional)

SearchOption As DWORD

A combination of the following flags:

0x00000000 SEARCH_SOURCE_OPTION_NONE
0x00000001 SEARCH_SOURCE_OPTION_USERDB

0x00000002 SEARCH_SOURCE_OPTION_USER_PERSONAL_ARCHIVE
0x00000004 SEARCH_SOURCE_OPTION_USER_EXTERNAL_ARCHIVE

QueryExecuteODMA()

Executes a named query returning the results to the ODMA dialog box.

Token ID

`AFTKN_QUERY_EXECUTE_ODMA or 831`

Syntax

```
VOID QueryExecuteODMA( [ANSISTRING QuickFinderQuery];
                       [DWORD Filter];
                       [ENUM WhereToSearch];
                       [ENUM OfficialVersion];
                       [ANSISTRING Libraries];
                       [ANSISTRING FullUserID];
                       [DWORD FolderListHandle];
                       [ANSISTRING FullUserID];
                       [DWORD FolderListHandle];
                       [ANSISTRING FullUserID];
                       [DWORD FolderListHandle];
                       [ANSISTRING FullUserID];
                       [DWORD FolderListHandle];
                       [ANSISTRING FullUserID];
                       [DWORD FolderListHandle];
                       [ANSISTRING FullUserID];
                       [DWORD FolderListHandle];
                       [ANSISTRING FullUserID];
                       [DWORD FolderListHandle];
                       [ANSISTRING FullUserID];
                       [DWORD FolderListHandle];
                       [ANSISTRING FullUserID];
                       [DWORD FolderListHandle];
                       [ANSISTRING FullUserID];
                       [DWORD FolderListHandle];
                       [ANSISTRING FullUserID];
                       [DWORD FolderListHandle];
                       [ANSISTRING FullUserID];
                       [DWORD FolderListHandle];
                       [ANSISTRING FullUserID];
                       [DWORD FolderListHandle];
                       [ANSISTRING FullUserID];
                       [DWORD FolderListHandle];
                       [ANSISTRING FullUserID];
                       [DWORD FolderListHandle];
                       [ANSISTRING FullUserID];
                       [DWORD FolderListHandle];
                       [ANSISTRING FullUserID];
                       [DWORD FolderListHandle];
                       [ANSISTRING FullUserID];
                       [DWORD FolderListHandle];
                       [ANSISTRING FullUserID];
                       [DWORD FolderListHandle];
                       [ANSISTRING FullUserID];
                       [DWORD FolderListHandle])
```

Parameters

QuickFinderQuery As ANSISTRING

(Optional) Name of the query.

Filter As DWORD

(Optional) Filter associated with the query.

WhereToSearch As ENUM

(Optional) Enumerated values:

2 RemoteMailBox
4 MasterMailBox
8 NNTP
16 IMAP

OfficialVersion As ENUM

(Optional) Enumerated values:

654 Official Only

Libraries As ANSISTRING

(Optional) Separates multiple libraries by a comma. In the form: Library.PostOffice.Domain.

FullUserID As ANSISTRING

(Optional) ID of the main user.

FolderListHandle As DWORD

(Optional)

FullUserID As ANSISTRING

(Optional) ID of a proxy user.

FolderListHandle As DWORD

(Optional)

FullUserID As ANSISTRING

(Optional) ID of a proxy user.

FolderListHandle As DWORD

(Optional)

FullUserID As ANSISTRING

(Optional) ID of a proxy user.

FolderListHandle As DWORD

(Optional)

FullUserID As ANSISTRING

(Optional) ID of a proxy user.

FolderListHandle As DWORD

(Optional)

FullUserID As ANSISTRING

(Optional) ID of a proxy user.

FolderListHandle As DWORD

(Optional)

FullUserID As ANSISTRING

(Optional) ID of a proxy user.

FolderListHandle As DWORD

(Optional)

FullUserID As ANSISTRING

(Optional) ID of a proxy user.

FolderListHandle As DWORD

(Optional)

FullUserID As ANSISTRING

(Optional) ID of a proxy user.

FolderListHandle As DWORD

(Optional)

FullUserID As ANSISTRING

(Optional) ID of a proxy user.

FolderListHandle As DWORD

(Optional)

FullUserID As ANSISTRING

(Optional) ID of a proxy user.

FolderListHandle As DWORD

(Optional)

FullUserID As ANSISTRING

(Optional) ID of a proxy user.

FolderListHandle As DWORD

(Optional)

FullUserID As ANSISTRING

(Optional) ID of a proxy user.

FolderListHandle As DWORD

(Optional)

FullUserID As ANSISTRING

(Optional) ID of a proxy user.

FolderListHandle As DWORD

(Optional)

FullUserID As ANSISTRING

(Optional) ID of a proxy user.

FolderListHandle As DWORD

(Optional)

FullUserID As ANSISTRING

(Optional) ID of a proxy user.

FolderListHandle As DWORD

(Optional)

FullUserID As ANSISTRING

(Optional) ID of a proxy user.

FolderListHandle As DWORD

(Optional)

FullUserID As ANSISTRING

(Optional) ID of a proxy user.

FolderListHandle As DWORD

(Optional)

FullUserID As ANSISTRING

(Optional) ID of a proxy user.

FolderListHandle As DWORD

(Optional)

FullUserID As ANSISTRING

(Optional) ID of a proxy user.

FolderListHandle As DWORD

(Optional)

FullUserID As ANSISTRING

(Optional) ID of a proxy user.

FolderListHandle As DWORD

(Optional)

Remarks

The first pair of FullUserIDs and FolderListHandles are for the main user. Subsequent pairs are for different proxy individuals.

QuerySaveAsFolder()

Creates a folder to display the results of the specified query when the folder is opened.

Token ID

```
AFTKN_QUERY_SAVE_AS_FOLDER or 848
```

Syntax

```
VOID QuerySaveAsFolder( ANSISTRING FolderName;
                        ANSISTRING FolderDesc;
                        WORD FolderPosition;
                        [BOOLEAN InvokeOnOpen];
                        [ANSISTRING QuickFinderQuery];
                        [DWORD Filter];
                        [ENUM WhereToSearch];
                        [ENUM OfficialVersion];
                        [ANSISTRING Libraries];
                        [ANSISTRING FullUserId];
                        [DWORD FolderListHandle])
```

Parameters

FolderName As ANSISTRING

The full FolderName path. For example, User's Full Name\Cabinet\FolderName.

FolderDesc As ANSISTRING

A description of the folder.

FolderPosition As WORD

The position the folder will be created in folderlist. 0 is in the first postion under the folder specified. For example, User's Full Name\Cabinet\FolderName would place the folder as the first folder under the Cabinet folder.

InvokeOnOpen As BOOLEAN

(Optional)

QuickFinderQuery As ANSISTRING

(Optional)

Filter As DWORD

(Optional)

WhereToSearch As ENUM

(Optional)

4 MasterMailBox
2 RemoteMailBox

OfficialVersion As ENUM

(Optional)

0 No

1 Yes

Libraries As ANSISTRING

Separates multiple libraries by a comma. In the form: Library.PostOffice.Domain. (Optional)

FullUserID As ANSISTRING

Selects all Folders in the user's account. This parameter, or the FolderListHandle parameter, is required if you wish to search in any of the accounts folders for items. If you use this parameter, do not use the FolderListHandle parameter. (Optional)

FolderListHandle As DWORD

A list of a single or multiple folders. If you use this folder, do not use the FullUserID parameter. (Optional)

Remarks

The Find Results window must be showing for this token to work.

See Also

"EnvPrefFullName()" on page 286 to get user's full name
"AddressBookResolveFullName()" on page 78 to get user's full address

QuerySaveAsFolderDlg()

Creates a folder to display the results of the specified query when the folder is opened. The Find Results window must be showing for this token to work.

Token ID

```
DTKN_QUERY_SAVE_AS_FOLDER or 101
```

Syntax

```
VOID QuerySaveAsFolderDlg()
```

QuerySaveAsFolderEx()

Creates a query folder with additional options for search source.

Token ID

```
DTKN_QUERY_SAVE_AS_FOLDEREX or 902
```

Syntax

```
VOID QuerySaveAsFolder(
     ANSISTRING FolderName;
     ANSISTRING FolderDesc;
     WORD FOlderPosition;
     [BOOL] InvokeonOpen;
     [ANSISTRING] QuickFinderQuery;
     [DWORD] Filter;
     [ENUM] WhereToSearch;
     [ENUM] OfficialVersion;
     [ANSISTRING] Libraries;
     [ANSISTRING] FullUserId;
     [DWORD] FolderListHandle;
     [DWORD] SearchOptions)
```

Parameters

(Only documenting parameters different from QuerySaveAsFolder())

SearchOptions As DWORD

A combination of the following flags:

0x00000000 SEARCH_SOURCE_OPTIONS_NONE
0x00000001 SEARCH_SOURCE_OPTIONS_USERDB
0x00000002 SEARCH_SOURCE_OPTIONS_USER_PERSONAL_ARCHIVE
0x00000004 SEARCH_SOURCE_OPTIONS_USER_EXTERNAL_ARCHIVE

QuerySimple()

Display a dialog that enables the user to execute a simple query.

Token ID

```
DTKN_QUERY_SIMPLE or 142
```

Syntax

```
VOID QuerySimple()
```

QueryStop()

Stops searching for items and documents in GroupWise.

Token ID

```
BFTKN_QUERY_STOP or 473
```

Syntax

```
VOID QueryStop()
```

QuickCorrect()

Allows GroupWise to automatically check and correct spelling, as the user types. A message item view must be showing with the cursor, or focus in the subject or message fields.

Token ID

```
BFTKN_QUICK_CORRECT or 430
```

Syntax

```
VOID QuickCorrect()
```

QuickViewerBottom()

Displays the QuickViewer at the bottom of the Item List.

Token ID

```
BKTKN_QV_BOTTOM or 1226
```

Syntax

```
VOID QuickViewerBottom()
```

QuickViewerHide()

Hides the QuickViewer screen of the main browser.

Token ID

```
BFTKN_HIDE_LOOKER or 441
```

Syntax

```
VOID QuickViewerHide()
```

See Also

“ShowQuickLook()” on page 839

QuickViewerRight()

Displays the QuickViewer at the right of the Item List.

Token ID

```
BKTKN_QV_RIGHT or 1225
```

Syntax

```
VOID QuickViewerRight()
```

QvButtonBarHide()

Hides the QuickViewer toolbar.

Token ID

```
BFTKN_QVBTNBAR_HIDE 1134
```

Syntax

```
VOID QvButtonBarHide()
```

QvButtonBarShow()

Shows or hides the QuickViewer toolbar.

Token ID

```
BFTKN_QVBTNBAR_SHOW 1133
```

Syntax

```
VOID QvButtonBarShow([ENUM State])
```

Parameters

State as ENUM

(Optional) Specifies the following:

0 Off
1 On

Remarks

If the State parameter is not passed in, this command acts as a toggle.

Refresh()

Updates Mailbox, Sent Items, Trash, or Calendar views to display items after the view was opened.

Token ID

```
BFTKN_REFRESH or 175
```

Syntax

```
VOID Refresh([ENUM PromptIfQueryIncomplete])
```

Parameters

PromptIfQueryIncomplete As ENUM

(Optional) Enumerated Values:

0 No
1 Yes

See Also

“InfoUpdate()” on page 423

RemoteConnect()

Connects GroupWise Remote to the master system and updates the remote mailbox.

Token ID

```
BFTKN_REM_CONNECT or 162
```

Syntax

```
VOID RemoteConnect( [ENUM RequestItems];
                    [ENUM RequestFolders];
                    [ENUM RequestRules];
                    [ENUM RequestPersonalGroups];
                    [ENUM RequestAddressBook];
                    [ENUM RequestPublicGroups];
                    [ENUM RequestSysAddressBook];
                    [ENUM RequestPersonalAddressBooks];
                    [DWORD SysAddressBookFilter];
                    [ENUM RequestNewsItems];
                    [ENUM PrimeCache])
```

Parameters

RequestItems As ENUM

(Optional) Request items if no item request is pending. Enumerated values:

0 No
1 Yes

RequestFolders As ENUM

(Optional) Enumerated values:

0 No
1 Yes

RequestRules As ENUM

(Optional) Enumerated values:

0 No
1 Yes

RequestPersonalGroups As ENUM

(Optional) Enumerated values:

0 No
1 Yes

RequestAddressBook As ENUM

(Optional) Enumerated values:

0 No
1 Yes

RequestPublicGroups As ENUM

(Optional) Enumerated values:

0 No
1 Yes

RequestSysAddressBook As ENUM

(Optional) Enumerated values:

0 No
1 Yes

RequestPersonalAddressBooks As ENUM

(Optional) Enumerated values:

0 No
1 Yes

SysAddressBookFilter As DWORD

(Optional) Description of parameter.

RequestNewsItems As ENUM

(Optional) Enumerated values:

0 No
1 Yes

PrimeCache As ENUM

(Optional) Enumerated values:

0 No
1 Yes

RemoteCreateModemConnection()

Creates a modem connection.

Token ID

```
AFTKN_REM_CREATE_ASYNC_CONNECTION or 786
```

Syntax

```
VOID RemoteCreateModemConnection( ANSISTRING ConnectionName;
                          ANSISTRING PhoneNumber;
                          ANSISTRING GatewayLoginID;
                          ANSISTRING GatewayPassword;
                           [ANSISTRING ModemScriptFilePath];
                          [ENUM DisconnectMethod];
                          [WORD RedialAttempts];
                          [WORD RedialInterval])
```

Parameters

ConnectionName As ANSISTRING

User-defined name.

PhoneNumber As ANSISTRING

The phone number for the connection.

GatewayLoginID As ANSISTRING

GatewayPassword As ANSISTRING

Passowrd for logging in to the gateway.

ModemScriptFilePath As ANSISTRING

(Optional) Location for the modem script.

DisconnectMethod As ENUM

(Optional) Specifies the disconnect method:

300	AfterAllUpdates
301	AfterRequestsSent
0	Manual

RedialAttempts As WORD

Default = 5. (Optional)

RedialInterval As WORD

Number of minutes: Default = 1. (Optional)

RemoteCreateNetworkConnection()

Creates a network connection.

Token ID

```
AFTKN_REM_CREATE_DIRECT_CONNECTION or 787
```

Syntax

```
VOID RemoteCreateNetworkConnection( ANSISTRING ConnectionName;
                                    ANSISTRING PostOfficePath;
                                    [ENUM DisconnectMethod])
```

Parameters

ConnectionName As ANSISTRING

User defined.

PostOfficePath As ANSISTRING

Include full path.

DisconnectMethod As ENUM

Specifies the disconnect method:

300	AfterAllUpdates! disconnects after all items are sent and all requests are processed and retrieved to the remote mailbox.
301	AfterRequestsSent! disconnects after all items are sent to the master mailbox and all waiting responses are retrieved. The connection does not wait for the master system to process requests. Manual! disconnects when you manually terminate the connection
0	Manual

RemoteDeleteConnection()

Deletes a remote connection from the Connection dialog box.

Token ID

```
AFTKN_REM_DELETECONNECTION or 804
```

Syntax

```
VOID RemoteDeleteConnection( ANSISTRING ConnectionName)
```

Parameters

ConnectionName As ANSISTRING

The name of the remote connection.

See Also

“RemoteCreateNetworkConnection()” on page 713

RemoteDisconnect()

Disconnects GroupWise Remote from the master system, and closes the Connection Status window. Token fails if there is no connection.

Token ID

```
BFTKN_REM_DISCONNECT or 382
```

Syntax

```
VOID RemoteDisconnect()
```

RemoteGetPhoneNumber()

Returns the phone number of the default modem connection.

Token ID

```
AFTKN_REM_GETDEFAULTCONNECTNUMBER or 823
```

Syntax

```
ANSISTRING RemoteGetPhoneNumber()
```

Return Values

ANSISTRING. Phone Number.

RemoteGWCheck()

Launches GWCheck against a Remote, Caching, or Archive database.

Token ID

```
BFTKN_REM_GWCHECK or 1051
```

Syntax

```
VOID RemoteGWCheck()
```

RemoteHitTheRoadDlg()

Runs the Wizard to create a remote post office from the user's master post office.

Token ID

```
DTKN_HIT_THE_ROAD or 117
```

Syntax

```
VOID RemoteHitTheRoadDlg()
```

RemoteModemCommand()

Sends commands to a modem while a Remote connection is being initiated.

Token ID

```
AFTKN_REM_MODEMCOMMAND or 822
```

Syntax

```
ANSISTRING RemoteModemCommand(ENUM ModemCommand;
                              [ANSISTRING ModemData)]
```

Parameters

ModemCommand As ENUM

Specifies the modem command:

312	AllowLogging Turns logging to the Log Window "On" or "Off" (see RemoteViewConnectionLog() (page 734).) Use th e string parameter.
308	Break Sends a one second break command to the modem. Ignore the optional string parameter.
309	CharAvailable Returns True if characters are available, otherwise False. Ignore the optional string parameter.
310	Connected Returns True if connected, otherwise False. Ignore the optional string parameter.
206	Read Returns the charaters waiting at the modem (more than one Read! may be needed to return a desired string). Ignore the optional string parameter.
313	Settings Modifies a modem's baud rate, parity, data bits, and stop bits, using a comma delimited string. An empty field uses a current setting. Use the string parameter.
311	TerminalMode Displays the Modem Terminal dialog box. Ignore the optional string parameter.
314	Write Sends the string parameter to the modem. Use the string parameter.

ModemData As ANSISTRING

(Optional) String data passed to a modem.

Return Values

ANSISTRING. Information about the modem connection.

RemotePendingRequestsDlg()

Displays the Pending Requests to Master Mailbox dialog box.

Token ID

```
DTKN_REM_UPDATE_INFO or 105
```

Syntax

```
VOID RemotePendingRequestsDlg()
```

RemoteRequestContacts()

Retrieves contacts into the remote database.

Token ID

```
BFTKN_REM_REQUEST_CONTACTS 1147
```

Syntax

```
VOID RemoteRequestContacts()
```

RemoteRequestDocument()

Requests the document specified by DocIDStr to be downloaded from the Master System to the Remote post office. The document will be marked In Use, based on the value of MarkInUse.

Token ID

```
AFTKN_REM_REQUEST_DOC or 878
```

Syntax

```
VOID RemoteRequestDocument( ANSISTRING DocIDStr;
                            ENUM MarkInUse)
```

Parameters

DocIDStr As ANSISTRING

Document ID for the document to download.

MarkInUse As ENUM

0 No
1 Yes

RemoteSelectedRetrieveDlg()

Displays the Retrieve Selected Items dialog box.

Token ID

```
DTKN_REM_SELECTEDUPDATE or 103
```

Syntax

```
VOID RemoteSelectedRetrieveDlg()
```

RemoteSendRetrieveDlg()

Displays the Send/Retrieve dialog box.

Token ID

```
DTKN_REM_UPDATE or 102
```

Syntax

```
VOID RemoteSendRetrieveDlg()
```

RemoteSetAddrBookDnloadFilter()

Requests user IDs, resources, and public group names from the master system.

Token ID

```
AFTKN_REM_SET_ADDRBOOK_DNLOAD_FILTER or 801
```

Syntax

```
VOID RemoteSetAddrBookDnloadFilter( [ANSISTRING DomainName];
                                    [ANSISTRING PostOfficeName])
```

Parameters

DomainName As ANSISTRING

(Optional) All post offices in this domain. If blank, all domains.

PostOfficeName As ANSISTRING

(Optional) One post office in a specified domain. Each post office represents a collection of mailboxes. If blank, all post offices.

See Also

“RemoteCreateNetworkConnection()” on page 713

RemoteSetDefaultConnection()

Sets the current modem or network connection.

Token ID

```
AFTKN_REM_SELECTCONNECTION or 797
```

Syntax

```
VOID RemoteSetDefaultConnection( ANSISTRING ConnectionName)
```

Parameters

ConnectionName As ANSISTRING

RemoteSetItemsFolders()

Downloads one or all folders from the master system. Repeat this token to add more than one token, but fewer than all.

Token ID

```
AFTKN_REM_SETITEMSFOLDERS or 800
```

Syntax

```
VOID RemoteSetItemsFolders( ENUM RetrieveContents;
                            [ANSISTRING FolderName])
```

Parameters

RetrieveContents As ENUM

Specifies what folders to retrieve:

287	Add
	Adds one folder to the list.
7	All
286	Reset
	Clears a previous folder list and adds one folder (see FolderName parameter) to a new folder list.

FolderName As ANSISTRING

Ignored if RetrieveContents is set to All. (Optional)

See Also

“RemoteCreateNetworkConnection()” on page 713

RemoteSetPreferences()

RemoteSetPreferences() sets remote preferences, such as name, user ID, etc.

Token ID

```
AFTKN_REM_SETPREFS or 654
```

Syntax

```
VOID RemoteSetPreferences( [ANSISTRING FullName];
                           [ANSISTRING UserID];
                           [ANSISTRING MMPassword];
                           [ANSISTRING Domain];
                           [ANSISTRING PostOffice];
                           [ANSISTRING Modem])
```

Parameters

FullName As ANSISTRING

(Optional) Full name of the user.

UserID As ANSISTRING

(Optional) GroupWise user ID of the user.

MMPassword As ANSISTRING

Master mailbox password. (Optional)

Domain As ANSISTRING

Post office location. (Optional)

PostOffice As ANSISTRING

Mailbox location. (Optional)

Modem As ANSISTRING

(Optional)

RemoteSetPublicGroupDnloadFilter()

Sets a filter for retrieving public groups.

Token ID

`AFTKN_REM_REQUESTPGROUP or 802`

Syntax

```
VOID RemoteSetPublicGroupDnloadFilter( ANSISTRING GroupName)
```

Parameters

GroupName As ANSISTRING

Name of the public group to download.

RemoteSetRequestItemsFilter()

Sets filters for retrieving items from the master system. Parameters not specified are not retrieved to the user's remote database.

Token ID

```
AFTKN_REM_SETITEMSFILTER or 799
```

Syntax

```
VOID RemoteSetRequestItemsFilter( [ENUM FromInbox];
                          [ENUM FromOutbox];
                          [ENUM FromPersonalBox];
                          [ENUM RetrieveMailAndPhone];
                          [ENUM RetrieveAppointments];
                          [ENUM RetrieveTasks];
                          [ENUM RetrieveNotes];
                          [DWORD MessageSizeLimit];
                          [DWORD AttachmentSizeLimit];
                          [DWORD DistributionListSizeLimit];
                          [WORD DaysPrior];
                          [WORD DaysAfter];
                          [ENUM RetrieveDocReferences])
```

Parameters

FromInbox As ENUM

(Optional)

0 No
1 Yes

FromOutbox As ENUM

(Optional)

0 No
1 Yes

FromPersonalBox As ENUM

(Optional)

0 No
1 Yes

RetrieveMailAndPhone As ENUM

(Optional) Specifies the mail and phone messages to retrieve:

7	All
0	None
280	Opened
283	Unopened

RetrieveAppointments As ENUM

(Optional) Specifies the appointments to retrieve:

104	Accepted
7	All
0	None
284	Unaccepted

RetrieveTasks As ENUM

(Optional) Specifies the tasks to retrieve:

7	All
138	Completed
0	None
285	Uncompleted

RetrieveNotes As ENUM

(Optional) Specifies the notes to retrieve:

7	All
0	None
280	Opened
283	Unopened

MessageSizeLimit As DWORD

Retrieve message text if less than a specified size in bytes. For example, 30720 (30x1024 or 30 kilobytes). (Optional)

AttachmentSizeLimit As DWORD

Retrieve attachments if less than a specified size in bytes. For example, 30720 (30x1023 or 30 kilobytes). (Optional)

DistributionListSizeLimit As DWORD

Retrieve To, CC, and BC text if less than a specified size in bytes. For example, 5120 (5x1024 or 5 kilobytes). (Optional)

DaysPrior As WORD

Retrieve items no older than a specified number of days prior to the current date. For example, 5. (Optional)

DaysAfter As WORD

Retrieve items that were received no later than a specified number of days after the current date. For example, 365. (Optional)

RetrieveDocReferences As ENUM

(Optional) Specifies the document references to retrieve:

7	All
0	None

RemoteViewConnectionLog()

Displays the Connection Log message box.

Token ID

`BFTKN_REM_VIEWCONNECTIONLOG or 381`

Syntax

```
VOID RemoteViewConnectionLog()
```

Rename()

Allows the user to rename the selected folder.

Token ID

```
BFTKN_RENAME or 229
```

Syntax

```
VOID Rename()
```

ResetNNTPFolder()

Clears an NNTP newsgroup folder.

Token ID

```
BFTKN_RESET_NNTP_FOLDER or 1006
```

Syntax

```
VOID ResetNNTPFolder([ANSISTRING FolderName];
                     [BOOLEAN KeepMessages])
```

Parameters

FolderName As ANSISTRING

(Optional) Newsgroup to clear.

KeepMessages As BOOLEAN

(Optional) Specify the following:

TRUE Keep any downloaded messages
FALSE Delete all messages

Remarks

This token always deletes all the header information.

Retrieve()

Retrieves a text file into a message box.

Token ID

```
AFTKN_RETRIEVE or 643
```

Syntax

```
VOID Retrieve(ANSISTRING Filename;
                [ENUM FileType])
```

Parameters

Filename As ANSISTRING

The name of the file to retrieve.

FileType As ENUM

(Optional) Enumerated values:

354 ASCII
355 HTML

RetrieveFileDlg()

Displays the Retrieve dialog box.

Token ID

```
DTKN_RETRIEVE or 72
```

Syntax

```
VOID RetrieveFileDlg()
```

RetrieveMime()

Import an individual MIME file into GroupWise.

Token ID

```
AFTKN_RETRIEVE_MIME or 950
```

Syntax

```
VOID RetrieveMime(
     ANSISTRING Filename;
     [ENUM] BoxType;
     [ANSISTRING] FolderName;
     [BOOL] CreateFolder)
```

Parameters

Filename As ANSISTRING

The full path of the file import.

BoxType As ENUM

(Optional) One of the following values:

6 Inbox
7 Outbox
666 Draft
198 Personal

FolderName As ANSISTRING

The full path of the folder in GroupWise where the item will be stored.

CreateFolder As BOOL

If TRUE and the folder specified in FolderName does not exist, it will be created.

RSSSubscribeDlg()

Display a dialog allowing the user to subscribe to an RSS feed.

Token ID

```
DTKN_RSS_SUBSCRIBE_DLG or 148
```

Syntax

```
VOID RSSSubscribeDlg(
     [WIDESTRING] RSSURL;
     [WIDESTRING] FolderName;
     [WIDESTRING] Description)
```

Parameters

RSSURL As WIDESTRING

(Optional) The URL of the RSS feed to subscribe to.

FolderName As WIDESTRING

(Optional) The name of the folder that will be created in GroupWise.

Description As WIDESTRING

(Optional) The description of the folder.

RuleAddActionAccept()

Accepts a task or appointment when rule conditions are met.

Token ID

```
AFTKN_RULE_ACT_ACCEPT or 644
```

Syntax

```
VOID RuleAddActionAccept( [ANSISTRING UserID];
                          ANSISTRING RuleName;
                          [ANSISTRING Comment])
```

Parameters

UserID As ANSISTRING

(Optional)

RuleName As ANSISTRING

Name of the rule to add the action to.

Comment As ANSISTRING

(Optional) Comment to include with accepted task or appointment.

See Also

“RuleCreate()” on page 760
“RuleDelete()” on page 762

RuleAddActionArchive()

RuleAddActionArchive() archives an item when rule conditions are met.

Token ID

```
AFTKN_RULE_ACT_ARCHIVE or 803
```

Syntax

```
VOID RuleAddActionArchive( [ANSISTRING UserID];
                           ANSISTRING  RuleName)
```

Parameters

UserID As ANSISTRING

(Optional)

RuleName As ANSISTRING

Name of the rule to add the action to.

See Also

“RuleCreate()” on page 760
“RuleDelete()” on page 762

RuleAddActionDecline()

Declines a task or appointment when rule conditions are met.

Token ID

```
AFTKN_RULE_ACT_DECLINE or 645
```

Syntax

```
VOID RuleAddActionDecline( [ANSISTRING UserID];
                           ANSISTRING RuleName;
                           [ANSISTRING Comment])
```

Parameters

UserID As ANSISTRING

(Optional)

RuleName As ANSISTRING

Name of the rule to add the action to.

Comment As ANSISTRING

Comment to include with declined task or appointment. (Optional)

See Also

“RuleCreate()” on page 760
“RuleDelete()” on page 762

RuleAddActionDelegate()

Delegates a task, appointment, or note item when rule conditions are met.

Token ID

```
AFTKN_RULE_ACT_DELEGATE or 646
```

Syntax

```
VOID RuleAddActionDelegate( [ANSISTRING UserID];
                            ANSISTRING RuleName;
                            ANSISTRING To;
                            [ANSISTRING SenderComments];
                            [ANSISTRING RecipientComments])
```

Parameters

UserID As ANSISTRING

(Optional)

RuleName As ANSISTRING

Name of the rule to add the action to.

To As ANSISTRING

User ID of the person delegated a task, appointment, or note.

SenderComments As ANSISTRING

Comments to the person who sent the task, appointment, or note. (Optional)

RecipientComments As ANSISTRING

Comments to the person delegated the task, appointment, or note. (Optional)

See Also

“RuleCreate()” on page 760
“RuleDelete()” on page 762

RuleAddActionEmptyItem()

Deletes an item and then empties it from Trash when rule conditions are met.

Token ID

```
AFTKN_RULE_ACT_EMPTYITEM or 647
```

Syntax

```
VOID RuleAddActionEmptyItem( [ANSISTRING UserID];
                             ANSISTRING RuleName)
```

Parameters

UserID As ANSISTRING

(Optional)

RuleName As ANSISTRING

Name of the rule to add the action to.

Remarks

IMPORTANT: You cannot recover emptied items.

See Also

“RuleCreate()” on page 760
“RuleDelete()” on page 762

RuleAddActionForward()

Forwards an item when rule conditions are met. Forwarded items are sent as an attachment to a mail message.

Token ID

```
AFTKN_RULE_ACT_FORWARD or 648
```

Syntax

```
VOID RuleAddActionForward( [ANSISTRING UserID];
                           ANSISTRING RuleName;
                           [ANSISTRING To];
                           [ANSISTRING CC];
                           [ANSISTRING BC];
                           [ANSISTRING Subject];
                           [ANSISTRING Files];
                           [ANSISTRING Message];
                           [ANSISTRING From])
```

Parameters

UserID As ANSISTRING

(Optional)

RuleName As ANSISTRING

Name of the rule to add the action to.

To As ANSISTRING

User ID of the person to receive the forwarded item. (Optional)

CC As ANSISTRING

User ID of the person to receive a copy of the forwarded item. (Optional)

BC As ANSISTRING

User ID of the person to receive a blind copy of the forwarded item. (Optional)

Subject As ANSISTRING

(Optional) Subject for the forwarded item.

Files As ANSISTRING

Separate multiple attachments with a comma. (Optional)

Message As ANSISTRING

(Optional) Message text for the forwarded item.

From As ANSISTRING

Name of person forwarding the item. (Optional)

See Also

“RuleCreate()” on page 760
“RuleDelete()” on page 762

RuleAddActionLinkToFolder()

Links an item to a folder when rule conditions are met.

Token ID

```
AFTKN_RULE_ACT_LINKTOFOLDER or 649
```

Syntax

```
VOID RuleAddActionLinkToFolder( [ANSISTRING UserID];
                                 ANSISTRING RuleName;
                                 ANSISTRING FolderName)
```

Parameters

UserID As ANSISTRING

(Optional)

RuleName As ANSISTRING

Name of the rule to add the action to.

FolderName As ANSISTRING

Include full folder path.

See Also

“RuleCreate()” on page 760
“RuleDelete()” on page 762

RuleAddActionMarkPrivate()

Marks an item private when rule conditions are met.

Token ID

```
AFTKN_RULE_ACT_MARKPRIVATE or 650
```

Syntax

```
VOID RuleAddActionMarkPrivate( [ANSISTRING UserID];
                               ANSISTRING RuleName)
```

Parameters

UserID As ANSISTRING

(Optional)

RuleName As ANSISTRING

Name of the rule to add the action to.

See Also

“RuleCreate()” on page 760
“RuleDelete()” on page 762

RuleAddActionMarkRead()

Adds an action to the rule, specified by RuleName, marking the item being processed as Read.

Token ID

```
AFTKN_RULE_ACT_MARKREAD or 850
```

Syntax

```
VOID RuleAddActionMarkRead( [ANSISTRING UserID];
                            ANSISTRING RuleName)
```

Parameters

UserID As ANSISTRING

(Optional)

RuleName As ANSISTRING

Name of the rule to add the action to.

RuleAddActionMarkUnread()

Adds an action to the rule, specified by RuleName, marking the item being processed as Unread.

Token ID

```
AFTKN_RULE_ACT_MARKUNREAD or 851
```

Syntax

```
VOID RuleAddActionMarkUnread( [ANSISTRING UserID];
                              ANSISTRING RuleName)
```

Parameters

UserID As ANSISTRING

(Optional)

RuleName As ANSISTRING

Name of the rule to add the action to.

RuleAddActionMoveToFolder()

Moves an item to a folder when rule conditions are met.

Token ID

```
AFTKN_RULE_ACT_MOVETOFOLDER or 651
```

Syntax

```
VOID RuleAddActionMoveToFolder([ANSISTRING UserID];
                               ANSISTRING RuleName;
                               ANSISTRING FolderName)
```

Parameters

UserID As ANSISTRING

(Optional)

RuleName As ANSISTRING

Name of the rule to add the action to.

FolderName As ANSISTRING

Include full folder path. The syntax of the folder's path is Folder1\Folder2\Folder3.

See Also

“RuleCreate()” on page 760
“RuleDelete()” on page 762

RuleAddActionReply()

Replies to the sender of a mail, appointment, task, or note item when rule conditions are met.

Token ID

```
AFTKN_RULE_ACT_REPLY or 652
```

Syntax

```
VOID RuleAddActionReply( [ANSISTRING UserID];
                         ANSISTRING RuleName;
                         ENUM ReplyType;
                         [ANSISTRING CC];
                         [ANSISTRING BC];
                         [ANSISTRING Subject];
                         [ANSISTRING Files];
                         [ANSISTRING Message];
                         [ANSISTRING From])
```

Parameters

UserID As ANSISTRING

(Optional)

RuleName As ANSISTRING

Name of the rule to add the action to.

ReplyType As ENUM

Reply to sender only or to sender and all recipients of the same mail, appointment, task, or note item.

232 ToAll
234 ToSender

CC As ANSISTRING

User ID of the person to receive a copy of the item. (Optional)

BC As ANSISTRING

User ID of the person to receive a blind copy of the item. (Optional)

Subject As ANSISTRING

(Optional) Subject for the item reply.

Files As ANSISTRING

Separate multiple attachments with a comma. (Optional)

Message As ANSISTRING

(Optional) Message for the item reply.

From As ANSISTRING

Name of person replying to the item. (Optional)

See Also

“RuleCreate()” on page 760
“RuleDelete()” on page 762

RuleAddActionReplyWithText()

Adds an action to the rule specified by RuleName that replies to the item being processed with a text message.

Token ID

```
AFTKN_RULE_ACT_REPLY_WITH_TEXT or 852
```

Syntax

```
VOID RuleAddActionReplyWithText( [ANSISTRING UserID];
                                 ANSISTRING  RuleName;
                                 ENUM ReplyType;
                                 [ANSISTRING CC];
                                 [ANSISTRING BC];
                                 [ANSISTRING Subject];
                                 [ANSISTRING Files];
                                 [ANSISTRING Message];
                                 [ANSISTRING From])
```

Parameters

UserID As ANSISTRING

(Optional)

RuleName As ANSISTRING

Name of the rule to add the action to.

ReplyType As ENUM

Reply to sender only or to sender and all recipients of the same mail, appointment, task, or note item.

232 ToAll
234 ToSender

CC As ANSISTRING

User ID of the person to receive a copy of the item. (Optional)

BC As ANSISTRING

User ID of the person to receive a blind copy of the item. (Optional)

Subject As ANSISTRING

(Optional) Subject for the reply.

Files As ANSISTRING

Separate multiple attachments with a comma. (Optional)

Message As ANSISTRING

(Optional) Message for the reply.

From As ANSISTRING

Name of person replying to the item. (Optional)

See Also

“RuleCreate()” on page 760
“RuleDelete()” on page 762

RuleAddActionSendMail()

Sends a mail item when rule conditions are met.

Token ID

```
AFTKN_RULE_ACT_SENDMAIL or 653
```

Syntax

```
VOID RuleAddActionSendMail( [ANSISTRING UserID];
                            [ANSISTRING RuleName];
                            [ANSISTRING To];
                            [ANSISTRING CC];
                            [ANSISTRING BC];
                            [ANSISTRING Subject];
                            [ANSISTRING Files];
                            [ANSISTRING Message];
                            [ANSISTRING From];
                            [ANSISTRING ViewName])
```

Parameters

UserID As ANSISTRING

(Optional)

RuleName As ANSISTRING

Name of the rule to add the action to.

To As ANSISTRING

Recipient user ID. (Optional)

CC As ANSISTRING

User ID of the person to receive a copy of the item. (Optional)

BC As ANSISTRING

User ID of the person to receive a blind copy of the item. (Optional)

Subject As ANSISTRING

(Optional) Subject for the sent item.

Files As ANSISTRING

Separate multiple attachments with a comma. (Optional)

Message As ANSISTRING

(Optional) Message for the sent item.

From As ANSISTRING

Name of person replying to the item. (Optional)

ViewName As ANSISTRING

Name of a mail view—such as "Mail"— or the name of a custom mail view. (Optional)

See Also

“RuleCreate()” on page 760

RuleAddActionStopRules()

Adds an action to the rule specified by RuleName that stops rule processing for the item being processed.

Token ID

```
AFTKN_RULE_ACT_STOPRULES or 824
```

Syntax

```
VOID RuleAddActionStopRules( [ANSISTRING UserID];
                             ANSISTRING RuleName)
```

Parameters

UserID As ANSISTRING

(Optional)

RuleName As ANSISTRING

Name of the rule to add the action to.

RuleCreate()

Creates a rule that sets up conditions for performing an action on one or more items. This command must be followed with RuleAddAction commands.

Token ID

```
AFTKN_RULE_CREATE or 638
```

Syntax

```
VOID RuleCreate( [ANSISTRING UserID];
                 ANSISTRING RuleName;
                 ENUM Event;
                 [ANSISTRING FolderName];
                 ENUM FilterHandleType;
                 [ENUM AppointmentConflict])
```

Parameters

UserID As ANSISTRING

(Optional)

RuleName As ANSISTRING

Name of the new rule.

Event As ENUM

Rule criteria: Activate rule on one of the following events.

136	CloseFolder
158	Exit
162	FiledItem
185	NewItem
194	OpenFolder
224	Startup
240	UserActivated

FolderName As ANSISTRING

The folder name when an event is CloseFolder!, FiledItem!, or OpenFolder! (Optional)

FilterHandleType As ENUM

Unique filter identifier, returned by FilterCreate().

315 Default

AppointmentConflict As ENUM

(Optional) Specifies appointment conflicts:

147	DontCare
0	No
1	Yes

See Also

“RuleDelete()” on page 762

RuleDelete()

Deletes a rule.

Token ID

```
AFTKN_RULE_DELETE or 639
```

Syntax

```
VOID RuleDelete( [ANSISTRING UserID];
                 [ANSISTRING RuleName)]
```

Parameters

UserID As ANSISTRING

(Optional)

RuleName As ANSISTRING

The name of the rule to delete.

See Also

“RuleCreate()” on page 760

RuleExecute()

Runs a rule on selected Mailbox, Sent Items, Trash or Calendar items.

Token ID

```
AFTKN_EXECUTE_RULE or 630
```

Syntax

```
VOID RuleExecute(ANSISTRING RuleName)
```

Parameters

RuleName As ANSISTRING

Name of the rule to run.

RuleExists()

Checks to see if a rule already exists.

Token ID

```
AFTKN_RULE_EXISTS or 883
```

Syntax

```
BOOLEAN RuleExists( [ANSISTRING UserID];
                    ANSISTRING RuleName)
```

Parameters

UserID As ANSISTRING

Allows someone to test if a rule exists for a proxied user assuming that the user has edit-rule rights on the proxied mailbox. If this parameter is not specified, the token assumes it should use the main logged-in UserID for the test. (Optional)

RuleName As ANSISTRING

The name of the rule.

Return Values

BOOLEAN. Returns True if a rule exists whose name matches that indicated by the Rulename parameter, False otherwise.

RuleGetCount()

Returns the number of rules that a user has defined.

Token ID

```
AFTKN_RULE_GET_COUNT or 940
```

Syntax

```
WORD RuleGetCount([ANSISTRING] UserID)
```

Parameters

UserID As ANSISTRING

(Optional) The user id of a proxy user. If not specified then the count of the user's rules are returned. Otherwise, the count of the proxy user's rules are returned.

Return Values

WORD. The number of rules defined for a user.

RuleGetName()

Retrieve the name of a rule.

Token ID

```
AFTKN_RULE_GET_NAME or 941
```

Syntax

```
ANSISTRING RuleGetName(
     [ANSISTRING] UserID;
     WORD Index)
```

Parameters

UserID As ANSISTRING

(Optional) The user id of a proxy user.

Index As WORD

The 0-based index of the rule. This is a value from 0 to RuleGetCount().

Return Values

WORD. The number of rules defined for a user..

RuleGetPosition()

Retrieve the position of the rule within the execution order.

Token ID

```
AFTKN_RULE_GET_POSITION or 942
```

Syntax

```
WORD RuleGetPosition(
     [ANSISTRING] UserID;
     ANSISTRING RuleName)
```

Parameters

UserID As ANSISTRING

(Optional) The user id of a proxy user.

RuleName As ANSISTRING

The name of the rules as returned by RuleGetName().

Return Values

WORD. The 0-based index of the position of the rule in the execution order.

RuleListDlg()

Displays the Rules dialog box to list, create, edit, copy, delete and run rules.

Token ID

```
DTKN_RULE_LIST or 62
```

Syntax

```
VOID RuleListDlg()
```

RuleSetPosition()

Sets the position of the rule in the rule execution order.

Token ID

```
AFTKN_RULE_SET_POSITION or 943
```

Syntax

```
VOID RuleSetPosition(
     [ANSISTRING] UserID;
     ANSISTRING RuleName;
     WORD RulePos)
```

Parameters

UserID As ANSISTRING

(Optional) The user id of a proxy user.

RuleName As ANSISTRING

The name of the rule as returned by RuleGetName().

RulePos As WORD

The new 0-based index of the rule within the execution order.

S

This section contains information about the following tokens:

ScrollLeft()

Display the previous day in the Week Calendar view.

Token ID

```
BFTKN_SCROLL_LEFT or 287
```

Syntax

```
VOID ScrollLeft()
```

See Also

“ScrollNext()” on page 773
“ScrollPrior()” on page 774
“ScrollRight()” on page 775

ScrollNext()

Displays the first day after the last day in the Week Calendar view as the first day. For example, if the last day of the week is June 17, the first day of the week is the June 18 after this command.

Token ID

```
BFTKN_SCROLL_NEXT or 289
```

Syntax

```
VOID ScrollNext()
```

See Also

“ScrollLeft()” on page 772
“ScrollPrior()” on page 774
“ScrollRight()” on page 775

ScrollPrior()

Displays the day before the first day in the Week Calendar view as the last day. For example, if the first day of the week is June 17, the last day of the week is June 16, after this command.

Token ID

```
BFTKN_SCROLL_PRIOR or 290
```

Syntax

```
VOID ScrollPrior()
```

See Also

“ScrollLeft()” on page 772
“ScrollNext()” on page 773
“ScrollRight()” on page 775

ScrollRight()

Displays the next day in the Week Calendar view.

Token ID

```
BFTKN_SCROLL_RIGHT or 288
```

Syntax

```
VOID ScrollRight()
```

See Also

“ScrollLeft()” on page 772
“ScrollNext()” on page 773
“ScrollPrior()” on page 774

SecurityProperties()

Displays the security properties of the selected item.

Token ID

```
DTKN_SECURITY_PROPERTIES or 110
```

Syntax

```
VOID SecurityProperties()
```

SelectDown()

Moves the insertion point down one line and selects the text in between.

Token ID

```
BFTKN_SELECTDOWN or 235
```

Syntax

```
VOID SelectDown()
```

Remarks

The message body field (in the Message View and the Object API) has a limit of 32,000 characters. The Message View will stop accepting data after 32,000 characters.

SelectLeft()

Moves the insertion point one character to the left and selects the character.

Token ID

```
BFTKN_SELECTLEFT or 236
```

Syntax

```
VOID SelectLeft()
```

Remarks

The message body field (in the Message View and the Object API) has a limit of 32,000 characters. The Message View will stop accepting data after 32,000 characters.

SelectLeftWord()

Moves the insertion point to the beginnning of the previous word and selects the text in between.

Token ID

```
BFTKN_SELECTLEFTWORD or 237
```

Syntax

```
VOID SelectLeftWord()
```

Remarks

The message body field (in the Message View and the Object API) has a limit of 32,000 characters. The Message View will stop accepting data after 32,000 characters.

SelectPageDown()

Moves the insertion point from its current position to the first line of the next screen, and selects the text in between.

Token ID

```
BFTKN_SELECTPAGEDOWN or 238
```

Syntax

```
VOID SelectPageDown()
```

Remarks

The message body field (in the Message View and the Object API) has a limit of 32,000 characters. The Message View will stop accepting data after 32,000 characters.

SelectPageUp()

Moves the insertion point form its current position to the first line of the previous screen, and selects the text in between.

Token ID

```
BFTKN_SELECTPAGEUP or 239
```

Syntax

```
VOID SelectPageUp()
```

Remarks

The message body field (in the Message View and the Object API) has a limit of 32,000 characters. The Message View will stop accepting data after 32,000 characters.

SelectRight()

Moves the insertion point one character to the right and selects the character.

Token ID

```
BFTKN_SELECTRIGHT or 240
```

Syntax

```
VOID SelectRight()
```

Remarks

The message body field (in the Message View and the Object API) has a limit of 32,000 characters. The Message View will stop accepting data after 32,000 characters.

SelectRightWord()

Moves the insertion point to the beginning of the next word and selects the text in between.

Token ID

```
BFTKN_SELECTRIGHTWORD or 241
```

Syntax

```
VOID SelectRightWord()
```

Remarks

The message body field (in the Message View and the Object API) has a limit of 32,000 characters. The Message View will stop accepting data after 32,000 characters.

SelectTimezoneDlg()

Invokes the Select Time Zone dialog box.

Token ID

`DTKN_SELECT_TIMEZONE or 120`

Syntax

```
VOID SelectTimezoneDlg()
```

Remarks

Valid from only the create appointment view.

SelectToBegLine()

Moves the insertion point from its current position to the beginning of the line and selects the text in between.

Token ID

```
BFTKN_SELECTBEGLINE or 231
```

Syntax

```
VOID SelectToBegLine()
```

Remarks

The message body field (in the Message View and the Object API) has a limit of 32,000 characters. The Message View will stop accepting data after 32,000 characters.

SelectToBegText()

Moves the insertion point from its current position to the beginning of a message box, and selects the text in between.

Token ID

```
BFTKN_SELECTBEGTEXT or 232
```

Syntax

```
VOID SelectToBegText()
```

Remarks

The message body field (in the Message View and the Object API) has a limit of 32,000 characters. The Message View will stop accepting data after 32,000 characters.

SelectToEndLine()

Moves the insertion point from its current position to the end of a line, and selects the text in between.

Token ID

```
BFTKN_SELECTENDLINE or 233
```

Syntax

```
VOID SelectToEndLine()
```

Remarks

The message body field (in the Message View and the Object API) has a limit of 32,000 characters. The Message View will stop accepting data after 32,000 characters.

SelectToEndText()

Moves the insertion point from its current position to the end of a message box and selects the text in between.

Token ID

```
BFTKN_SELECTENDTEXT or 234
```

Syntax

```
VOID SelectToEndText()
```

Remarks

The message body field (in the Message View and the Object API) has a limit of 32,000 characters. The Message View will stop accepting data after 32,000 characters.

SelectUp()

Moves the insertion point up one line and selects the text in between.

Token ID

```
BFTKN_SELECTUP or 242
```

Syntax

```
VOID SelectUp()
```

Remarks

The message body field (in the Message View and the Object API) has a limit of 32,000 characters. The Message View will stop accepting data after 32,000 characters.

SendAppointment()

Schedules an appointment.

Token ID

```
AFTKN_SEND_APPT or 581
```

Syntax

```
DWORD SendAppointment( ANSISTRING To;
                       [ANSISTRING Subject];
                       [WORD StartDay];
                       [WORD StartMonth];
                       [WORD StartYear];
                       [WORD StartMinute];
                       [WORD StartHour];
                       [WORD EndDay];
                       [WORD EndMonth];
                       [WORD EndYear];
                       [WORD EndMinute];
                       [WORD EndHour];
                       [ANSISTRING Message];
                       [ANSISTRING Attach];
                       [ANSISTRING Place];
                       [ANSISTRING CC];
                       [ANSISTRING BC];
                       [ANSISTRING From];
                       [ANSISTRING ViewName];
                       [ENUM AttachListHasDisplayNames];
                       [ANSISTRING UserID];
                       [ANSISTRING MessageIDs];
                       [ENUM MessageIsFile];
                       [ENUM SubjectIsFile];
                       [ENUM ViewNameIsFile];
                       [ANSISTRING OleAttach];
                       [ANSISTRING SenderID])
```

Parameters

To As ANSISTRING

User ID of one or more appointment recipients. Separate multiple recipients with commas.

Subject As ANSISTRING

Parameter also accepts a file name. (Optional)

StartDay As WORD

(Optional)

StartMonth As WORD

(Optional)

StartYear As WORD

(Optional)

StartMinute As WORD

(Optional)

StartHour As WORD

(Optional)

EndDay As WORD

(Optional)

EndMonth As WORD

(Optional)

EndYear As WORD

(Optional)

EndMinute As WORD

(Optional)

EndHour As WORD

(Optional)

Message As ANSISTRING

Parameter also accepts a file name. (Optional)

Attach As ANSISTRING

Separate multiple attachments with commas. (Optional)

Place As ANSISTRING

(Optional)

CC As ANSISTRING

User ID of one or more carbon copy recipients. Separate multiple recipients with commas. (Optional)

BC As ANSISTRING

User ID of one or more blind copy recipients. Separate multiple recipients with commas. (Optional)

From As ANSISTRING

Name of person scheduling the appointment. (Optional)

ViewName As ANSISTRING

Name of an Appointment view, such as "Meeting w/attach", or the name of a custom Appointment view. Parameter also accepts a file name. (Optional)

AttachListHasDisplayNames As ENUM

(Optional)

0 No

1 Yes and the Attach parameters must include the file name and display name for each attachment, respectively.

UserID As ANSISTRING

Use this parameter to schedule an appointment from a proxy mailbox. (Optional)

MessageIDs As ANSISTRING

Attach encapsulated items. Separate multiple message IDs with a comma. (Optional)

MessageIsFile As ENUM

Message parameter contains a file name. (Optional)

0 No
1 Yes

SubjectIsFile As ENUM

Subject parameter contains a file name. (Optional)

0 No
1 Yes

ViewNameIsFile As ENUM

ViewName parameter contains a file name. (Optional)

0 No
1 Yes

OleAttach As ANSISTRING

(Optional)

SenderID As ANSISTRING

(Optional)

Return Values

IDforEnvSentMessageID DWORD. Staged message ID for tracking an appointment item. Use EnvSentMessageID() token to get actual message ID of the appointment.

Remarks

The message body field (in the Message View and the Object API) has a limit of 32,000 characters. The Message View will stop accepting data after 32,000 characters.

SendMail()

Schedules a mail item.

Token ID

```
AFTKN_SEND_MESSAGE or 578
```

Syntax

```
DWORD SendMail( ANSISTRING To;
                [ANSISTRING Subject];
                [ANSISTRING Message];
                [ANSISTRING Attach];
                [ANSISTRING CC];
                [ANSISTRING BC];
                [ANSISTRING From];
                [ENUM IsRouted];
                [ANSISTRING ViewName];
                [ENUM AttachListHasDisplayNames];
                [ANSISTRING UserID];
                [ANSISTRING MessageIDs];
                [ENUM MessageIsFile];
                [ENUM SubjectIsFile];
                [ENUM ViewNameIsFile];
                [ENUM AttachmentsArePersonal];
                [ANSISTRING OleAttach];
                [ANSISTRING SenderID])
```

Parameters

To As ANSISTRING

User ID of one or more appointment recipients. Separate multiple recipients with commas.

Subject As ANSISTRING

Parameter also accepts a file name. (Optional)

Message As ANSISTRING

Parameter also accepts a file name. (Optional)

Attach As ANSISTRING

Separate multiple attachments with commas. (Optional)

CC As ANSISTRING

User ID of one or more carbon copy recipients. Separate multiple recipients with commas. (Optional)

BC As ANSISTRING

User ID of one or more blind copy recipients. Separate multiple recipients with commas. (Optional)

From As ANSISTRING

Name of person scheduling the appointment. (Optional)

IsRouted As ENUM

Route mail item to two or more users in turn. (Optional)

0 No
1 Yes

ViewName As ANSISTRING

Name of a Mail view. For example, "Expanded Mail", or the name of a custom view. Parameter also accepts a file name. (Optional)

AttachListHasDisplayNames As ENUM

(Optional)

0 No
1 Yes and the Attach parameter must include the file name and display name for each attachment, respectively.

UserID As ANSISTRING

Use this parameter to schedule an appointment from a proxy mailbox. (Optional)

MessageIDs As ANSISTRING

Attach encapsulated items. Separate multiple message IDs with a comma. (Optional)

MessageIsFile As ENUM

Message parameter contains a file name. (Optional)

0 No
1 Yes

SubjectIsFile As ENUM

Subject parameter contains a file name. (Optional)

0 No
1 Yes

ViewNameIsFile As ENUM

ViewName parameter contains a file name. (Optional)

0 No
1 Yes

AttachmentsArePersonal As ENUM

(Optional)

0 No
1 Yes

OleAttach As ANSISTRING

(Optional)

SenderID As ANSISTRING

(Optional)

Return Values

IDforEnvSentMessageID DWORD. Staged message ID for tracking a mail item. Use EnvSentMessageID() token to get actual message ID of the mail item.

Remarks

The message body field (in the Message View and the Object API) has a limit of 32,000 characters. The Message View will stop accepting data after 32,000 characters.

SendNNTP()

Posts the current message to the given newsgroup.

Token ID

```
BFTKN_SEND_NNTP or 1007
```

Syntax

```
VOID SendNNTP([ANSISTRING FolderName])
```

Parameters

FolderName As ANSISTRING

(Optional) Newsgroup to post to.

Remarks

Depending on the server, this token can fail for various reasons.

SendNote()

Schedules a note item.

Token ID

```
AFTKN_SEND_NOTE or 579
```

Syntax

```
DWORD SendNote( ANSISTRING To;
                [ANSISTRING Subject];
                [WORD StartDay];
                [WORD StartMonth];
                [WORD StartYear];
                [ANSISTRING Note];
                [ANSISTRING Attach];
                [ANSISTRING CC];
                [ANSISTRING BC];
                [ANSISTRING From];
                [ANSISTRING ViewName];
                [ENUM AttachListHasDisplayNames];
                [ANSISTRING UserID];
                [ANSISTRING MessageIDs];
                [ENUM MessageIsFile];
                [ENUM SubjectIsFile];
                [ENUM ViewNameIsFile];
                [ANSISTRING OleAttach];
                [ANSISTRING SenderID])
```

Parameters

To As ANSISTRING

User ID of one or more note recipients. Separate multiple recipients with commas.

Subject As ANSISTRING

Parameter also accepts a file name. (Optional)

StartDay As WORD

(Optional)

StartMonth As WORD

(Optional)

StartYear As WORD

(Optional)

Note As ANSISTRING

Parameter also accepts a file name. (Optional)

Attach As ANSISTRING

Separate multiple attachments with commas. (Optional)

CC As ANSISTRING

User ID of one or more note recipients. Separate multiple recipients with commas.

BC As ANSISTRING

User ID of one or more blind copy recipients. Separate multiple recipients with commas. (Optional)

From As ANSISTRING

Name of person sending the notice. (Optional)

ViewName As ANSISTRING

Name of a Note view. For example, "Notice w/attach", or the name of a custom Note view. Parameter also accepts a file name. (Optional)

AttachListHasDisplayNames As ENUM

(Optional)

0 No

1 Yes and the Attach parameter must include the file name and display name for each attachment, respectively.

UserID As ANSISTRING

Use this parameter to send a note item from a proxy mailbox. (Optional)

MessageIDs As ANSISTRING

Attach encapsulated items. Separate multiple message IDs with a comma. (Optional)

MessageIsFile As ENUM

Message parameter contains a file name. (Optional)

0 No

1 Yes

SubjectIsFile As ENUM

Subject parameter contains a file name. (Optional)

0 No

1 Yes

ViewNameIsFile As ENUM

ViewName parameter contains a file name. (Optional)

0 No

1 Yes

OleAttach As ANSISTRING

(Optional)

SenderID As ANSISTRING

(Optional)

Return Values

IDforEnvSentMessageID DWORD. Staged message ID for tracking a note item. Use EnvSentMessageID() token to get actual message ID of the note item.

Remarks

The message body field (in the Message View and the Object API) has a limit of 32,000 characters. The Message View will stop accepting data after 32,000 characters.

SendOptions()

Sets Send Options for the current item (see SendMail(), for example). Not all options are available for every item.

Token ID

```
AFTKN_SEND_OPTIONS or 636
```

Syntax

```
DWORD SendOptions( [ENUM Priority];
                   [ENUM Notify];
                   [ENUM ReturnOnOpen];
                   [ENUM ReturnOnAccept];
                   [ENUM ReturnOnCompleted];
                   [ENUM ReturnOnDelete];
                   [ENUM StatusInfo];
                   [ENUM ReplyRequested];
                   [WORD ReplyRequestedDays];
                   [WORD ExpireDays];
                   [ENUM AutoDelete])
```

Parameters

Priority As ENUM

(Optional) Specifies the priority:

0	Low
1	Normal
2	High

Notify As ENUM

Notify recipients when an item arrives. (Optional)

0 No
1 Yes

ReturnOnOpen As ENUM

Return notification options. (Optional)

276	Mail
3	MailandNotify
0	None
2	Notify
1	xOldMail

ReturnOnAccept As ENUM

Return notifications options. (Optional)

276	Mail
3	MailandNotify
0	None
2	Notify
1	xOldMail

ReturnOnCompleted As ENUM

Return notification options. (Optional)

276	Mail
3	MailandNotify
0	None
2	Notify
1	xOldMail

ReturnOnDelete As ENUM

Return notification options. (Optional)

276	Mail
3	MailandNotify
0	None
2	Notify
1	xOldMail

StatusInfo As ENUM

Information to track when a mail or phone item is sent. (Optional)

3	AllInfo
275	Delivered
1	Delivery
0	None
193	Open
280	Opened

ReplyRequested As ENUM

Request reply on the first line of the message box. (Optional)

0	None
1	WhenConvenient
2	WithinDays

ReplyRequestedDays As WORD

Number of days. The recipients sees: Reply Requested: By xx/xx/xx. (Optional)

ExpireDays As WORD

Number of days a mail or phone message remains in a recipient's Mailbox. (Optional)

AutoDelete As ENUM.

Delete the item from the sender's Sent Items when it is deleted from the recipient's Mailbox. (Optional)

0 No
1 Yes

Return Values

IDforEnvSentMessageID DWORD. Staged message ID for tracking a note item. Use EnvSentMessageID() token to get actual message ID of the note item.

See Also

"PrefAdvanced()" on page 641
"SendOptionsDlg()" on page 803

SendOptionsDlg()

Displays the Send Options dialog box of the current item. For example, for task items, this command displays the Task Send Options dialog box.

Token ID

```
DTKN_SEND_OPTIONS or 47
```

Syntax

```
VOID SendOptionsDlg()
```

See Also

“SendOptions()” on page 800

SendPhone()

Sends a phone item.

Token ID

```
AFTKN_SEND_PHONE or 604
```

Syntax

```
DWORD SendPhone( ANSISTRING To;
                 [ANSISTRING CallerName];
                 [ANSISTRING CallerCompanyName];
                 [ANSISTRING CallerPhone];
                 [ENUM Telephoned];
                 [ENUM PleaseCall];
                 [ENUM WillCallAgain];
                 [ENUM ReturnedYourCall];
                 [ENUM WantsToSeeYou];
                 [ENUM CameToSeeYou];
                 [ENUM Urgent];
                 [ANSISTRING Subject];
                 [ANSISTRING Message];
                 [ANSISTRING Attach];
                 [ANSISTRING CC];
                 [ANSISTRING BC];
                 [ANSISTRING From];
                 [ANSISTRING ViewName];
                 [ENUM AttachListHasDisplayNames];
                 [ANSISTRING UserID];
                 [ANSISTRING MessageIDs];
                 [ENUM MessageIsFile];
                 [ENUM SubjectIsFile];
                 [ENUM ViewNameIsFile];
                 [ANSISTRING OleAttach];
                 [ANSISTRING SenderID])
```

Parameters

To As ANSISTRING

User ID of phone item recipients. Separate multiple recipients with commas.

CallerName As ANSISTRING

(Optional)

CallerCompanyName As ANSISTRING

(Optional)

CallerPhone As ANSISTRING

(Optional)

Telephoned As ENUM

(Optional)

0 No
1 Yes

PleaseCall As ENUM

Return phone call. (Optional)

0 No
1 Yes

WillCallAgain As ENUM

(Optional)

0 No
1 Yes

ReturnedYourCall As ENUM

(Optional)

0 No
1 Yes

WantsToSeeYou As ENUM

(Optional)

0 No
1 Yes

CameToSeeYou As ENUM

(Optional)

0 No
1 Yes

Urgent As ENUM

(Optional)

0 No
1 Yes

Subject As ANSISTRING

Parameter also accepts a file name. (Optional)

Message As ANSISTRING

Parameter also accepts a file name. (Optional)

Attach As ANSISTRING

Separate multiple attachments with commas. (Optional)

CC As ANSISTRING

User ID of one or more carbon copy recipients. Separate multiple recipients with commas. (Optional)

BC As ANSISTRING

User ID of one or more blind copy recipients. Separate multiple recipients with commas. (Optional)

From As ANSISTRING

Name of person scheduling the appointment. (Optional)

ViewName As ANSISTRING

(Optional)

AttachListHasDisplayNames As ENUM

(Optional)

0 No
1 Yes

UserID As ANSISTRING

Use this parameter to schedule an appointment from a proxy mailbox. (Optional)

MessageIDs As ANSISTRING

Attach encapsulated items. Separate multiple message IDs with a comma. (Optional)

MessageIsFile As ENUM

Message parameter contains a file name. (Optional)

0 No
1 Yes

SubjectIsFile As ENUM

Subject parameter contains a file name. (Optional)

0 No
1 Yes

ViewNameIsFile As ENUM

ViewName parameter contains a file name. (Optional)

0 No
1 Yes

OleAttach As ANSISTRING

(Optional)

SenderID As ANSISTRING

(Optional)

Return Values

IDforEnvSentMessageID DWORD. Staged message ID for tracking a mail item. Use EnvSentMessageID() token to get actual message ID of the mail item.

Remarks

The message body field (in the Message View and the Object API) has a limit of 32,000 characters. The Message View will stop accepting data after 32,000 characters.

See Also

“CheckboxSelect()” on page 146

SendTask()

Sends a task item.

Token ID

```
AFTKN_SEND_TODO or 580
```

Syntax

```
DWORD SendTask( ANSISTRING To;
          [ANSISTRING Subject];
          [ANSISTRING CategoryAndPriority];
          [WORD StartDay];
          [WORD StartMonth];
          [WORD StartYear];
          [WORD DueDay];
          [WORD DueMonth];
          [WORD DueYear];
          [ANSISTRING Message];
          [ANSISTRING Attach];
          [ANSISTRING CC];
          [ANSISTRING BC];
          [ANSISTRING From];
          [ENUM IsRouted];
          [ANSISTRING ViewName];
          [ENUM AttachListHasDisplayNames];
          [ANSISTRING UserID];
          [ANSISTRING MessageIDs];
          [ENUM MessageIsFile];
          [ENUM SubjectIsFile];
          [ENUM ViewNameIsFile];
          [ENUM AttachmentsArePersonal];
          [ANSISTRING OleAttach];
          [ANSISTRING SenderID])
```

Parameters

To As ANSISTRING

User ID of task recipients. Separate multiple recipients with commas.

Subject As ANSISTRING

Parameter also accepts a file name. (Optional)

CategoryAndPriority As ANSISTRING

(Optional)

StartDay As WORD

(Optional)

StartMonth As WORD

(Optional)

StartYear As WORD

(Optional)

DueDay As WORD

(Optional)

DueMonth As WORD

(Optional)

DueYear As WORD

(Optional)

Message As ANSISTRING

Parameter also accepts a file name. (Optional)

Attach As ANSISTRING

Separate multiple attachments with commas. (Optional)

CC As ANSISTRING

User ID of one or more carbon copy recipients. Separate multiple recipients with commas. (Optional)

BC As ANSISTRING

User ID of one or more blind copy recipients. Separate multiple recipients with commas. (Optional)

From As ANSISTRING

Name of person scheduling the appointment. (Optional)

IsRouted As ENUM

Route task item to two or more users in turn.

0 No
1 Yes

ViewName As ANSISTRING

Name of a Mail view, such as "Expanded Mail", or the name of a custom Mail view. Parameter also accepts a file name. (Optional)

AttachListHasDisplayNames As ENUM

(Optional)

0 No
1 Yes

UserID As ANSISTRING

Use this parameter to schedule an appointment from a proxy mailbox. (Optional)

MessageIDs As ANSISTRING

Attach encapsulated items. Separate multiple message IDs with a comma. (Optional)

MessageIsFile As ENUM

Message parameter contains a file name. (Optional)

0 No
1 Yes

SubjectIsFile As ENUM

Subject parameter contains a file name. (Optional)

0 No
1 Yes

ViewNameIsFile As ENUM

ViewName parameter contains a file name. (Optional)

0 No
1 Yes

AttachmentsArePersonal As ENUM

(Optional)

0 No
1 Yes

OleAttach As ANSISTRING

(Optional)

SenderID As ANSISTRING

(Optional)

Return Values

IDforEnvSentMessageID DWORD. Staged message ID for tracking a mail item. Use EnvSentMessageID() token to get actual message ID of the mai item.

Remarks

The message body field (in the Message View and the Object API) has a limit of 32,000 characters. The Message View will stop accepting data after 32,000 characters.

See Also

“CheckboxSelect()” on page 146

SendURL()

Creates a Mail message with http:// pre-added to the subject line.

Token ID

```
BFTKN_SEND_URL or 476
```

Syntax

```
VOID SendURL()
```

See Also

“SendOptions()” on page 800

SetCalDisplayOptions()

Sets the Date Time Options dialog preference items. There is no refresh, so changes are not readily apparent upon issueing the token.

Token ID

```
AFTKN_SET_CAL_DISPLAY_OPTIONS or 889
```

Syntax

```
VOID SetCalDisplayOptions( [ENUM ApptColor];
                           [ENUM NoteColor];
                           [ENUM TodoColor];
                           [BOOLEAN Unused];
                           [ENUM NoteLines];
                           [ENUM TodoLines])
```

Parameters

ApptColor As ENUM

(Optional)

0 No
1 Yes

NoteColor As ENUM

(Optional)

0 No
1 Yes

TodoColor As ENUM

(Optional)

0 No
1 Yes

Unused As BOOLEAN

(Optional)

NoteLines As ENUM

(Optional)

0 No
1 Yes

TodoLines As ENUM

(Optional)

0 No
1 Yes

SetColumns()

Sets the columns and their associated properties for display in the GroupWise client.

Token ID

```
AFTKN_SET_COLUMNS or 846
```

Syntax

```
VOID SetColumns( [ENUM Column00]
                 [ANSISTRING ColumnFieldName00];
                 [WORD ColumnWidth00];
                 [ENUM Column01];
                 [ANSISTRING ColumnFieldName01];
                 [WORD ColumnWidth01];
                 [ENUM Column02];
                 [ANSISTRING ColumnFieldName02];
                 [WORD ColumnWidth02];
                 [ENUM Column03];
                 [ANSISTRING ColumnFieldName03];
                 [WORD ColumnWidth03];
                 [ENUM Column04];
                 [ANSISTRING ColumnFieldName04];
                 [WORD ColumnWidth04];
                 [ENUM Column05];
                 [ANSISTRING ColumnFieldName05];
                 [WORD ColumnWidth05];
                 [ENUM Column06];
                 [ANSISTRING ColumnFieldName06];
                 [WORD ColumnWidth06];
                 [ENUM Column07];
                 [ANSISTRING ColumnFieldName07];
                 [WORD ColumnWidth07];
                 [ENUM Column08];
                 [ANSISTRING ColumnFieldName08];
                 [WORD ColumnWidth08];
                 [ENUM Column09];
                 [ANSISTRING ColumnFieldName09];
                 [WORD ColumnWidth09];
                 [ENUM Column10];
                 [ANSISTRING ColumnFieldName10];
                 [WORD ColumnWidth10];
                 [ENUM Column11];
                 [ANSISTRING ColumnFieldName11];
                 [WORD ColumnWidth11];
                 [ENUM Column12];
                 [ANSISTRING ColumnFieldName12];
                 [WORD ColumnWidth12];
                 [ENUM Column13];
                 [ANSISTRING ColumnFieldName13];
                 [WORD ColumnWidth13];
                 [ENUM Column14];
                 [ANSISTRING ColumnFieldName14];
                 [WORD ColumnWidth14];
                 [ENUM Column15];
                 [ANSISTRING ColumnFieldName15];
```

```
[WORD ColumnWidth15];
[ENUM Column16];
[ANSISTRING ColumnFieldName16];
[WORD ColumnWidth16];
[ENUM Column17];
[ANSISTRING ColumnFieldName17];
[WORD ColumnWidth17];
[ENUM Column18];
[ANSISTRING ColumnFieldName18];
[WORD ColumnWidth18];
[ENUM Column19];
[ANSISTRING ColumnFieldName19];
[WORD ColumnWidth19])
```

Parameters

Column00 As ENUM

(Optional) Enumerated values:

Number	Value
600	AcceptedCnt
112	AssignedDate
625	Author
4	Authority
126	BeginDateTime
6	Caller
7	Company
603	CompletedCut
602	CopyType
139	CreateDate
626	Creator
176	Date
606	DeletedCnt
605	DeletedDate
507	DeliveredDate
656	DocCreatedDate
608	DocumentID
629	DocumentType
148	DueDate
151	EndDateTime
611	Folder

Number	Value
1	From
169	ItemType
614	Library
647	LibraryId
633	Modified
184	Name
331	NamedField
703	NamedFldByte
702	NamedFldDate
706	NamedFldSDWord
704	NamedFldSWord
700	NamedFldText
707	NamedFldUDWord
705	NamedFldUWord
701	NamedFldWrdStr
615	OpenedCnt
8	Phone
5	Place
616	RecipientsCnt
617	RepliedCnt
657	SharedName
618	Size
619	Source
646	Subclass
9	Subject
227	TaskCategory
230	TaskPriority
351	ThreadStatus
0	To
622	UndelieveredCnt
623	Version
634	VersionCreator
635	VersionDate

Number	Value
636	VersionDescription
267	ViewName

ColumnFieldName00 As ANSISTRING

(Optional)

ColumnWidth00 As WORD

(Optional)

Column01 As ENUM

(Optional) Same values as for Column00.

ColumnFieldName01 As ANSISTRING

(Optional)

ColumnWidth01 As WORD

(Optional)

Column02 As ENUM

(Optional) Same values as for Column00

ColumnFieldName02 As ANSISTRING

(Optional)

ColumnWidth02 As WORD

(Optional)

Column03 As ENUM

(Optional) Same values as for Column00

ColumnFieldName03 As ANSISTRING

(Optional)

ColumnWidth03 As WORD

(Optional)

Column04 As ENUM

(Optional) Same values as for Column00

ColumnFieldName04 As ANSISTRING

(Optional)

ColumnWidth04 As WORD

(Optional)

Column05 As ENUM

(Optional) Same values as for Column00

ColumnFieldName05 As ANSISTRING

(Optional)

ColumnWidth05 As WORD

(Optional)

Column06 As ENUM

(Optional) Same values as for Column00

ColumnFieldName06 As ANSISTRING

(Optional)

ColumnWidth06 As WORD

(Optional)

Column07 As ENUM

(Optional) Same values as for Column00

ColumnFieldName07 As ANSISTRING

(Optional)

ColumnWidth07 As WORD

(Optional)

Column08 As ENUM

(Optional) Same values as for Column00

ColumnFieldName08 As ANSISTRING

(Optional)

ColumnWidth08 As WORD

(Optional)

Column09 As ENUM

(Optional) Same values as for Column00

ColumnFieldName09 As ANSISTRING

(Optional)

ColumnWidth09 As WORD

(Optional)

Column10 As ENUM

(Optional) Same values as for Column00

ColumnFieldName10 As ANSISTRING

(Optional)

ColumnWidth10 As WORD

(Optional)

Column11 As ENUM

(Optional) Same values as for Column00

ColumnFieldName11 As ANSISTRING

(Optional)

ColumnWidth11 As WORD

(Optional)

Column12 As ENUM

(Optional) Same values as for Column00

ColumnFieldName12 As ANSISTRING

(Optional)

ColumnWidth12 As WORD

(Optional)

Column13 As ENUM

(Optional) Same values as for Column00

ColumnFieldName13 As ANSISTRING

(Optional)

ColumnWidth13 As WORD

(Optional)

Column14 As ENUM

(Optional) Same values as for Column00

ColumnFieldName14 As ANSISTRING

(Optional)

ColumnWidth14 As WORD

(Optional)

Column15 As ENUM

(Optional) Same values as for Column00

ColumnFieldName15 As ANSISTRING

(Optional)

ColumnWidth15 As WORD

(Optional)

Column16 As ENUM

(Optional) Same values as for Column00

ColumnFieldName16 As ANSISTRING

(Optional)

ColumnWidth16 As WORD

(Optional)

Column17 As ENUM

(Optional) Same values as for Column00

ColumnFieldName17 As ANSISTRING

(Optional)

ColumnWidth17 As WORD

(Optional)

Column18 As ENUM

(Optional) Same values as for Column00

ColumnFieldName18 As ANSISTRING

(Optional)

ColumnWidth18 As WORD

(Optional)

Column19 As ENUM

(Optional) Same values as for Column00

ColumnFieldName19 As ANSISTRING

(Optional)

ColumnWidth19 As WORD

(Optional)

SetDay()

Sets the first day of the week, or which day of the week will be the first column of the month control.

Token ID

```
AFTKN_SET_DAY or 869
```

Syntax

```
VOID SetDay( ENUM FirstDayOfWeek)
```

Parameters

FirstDayofWeek As ENUM

Specifies the first day of the week:

5	Friday
1	Monday
6	Saturday
0	Sunday
4	Thursday
2	Tuesday
3	Wednesday

SetMonthDisplayOptions()

Sets the global display options for the Month calendar control.

Token ID

```
AFTKN_SET_MONTH_DISPLAY_OPTIONS or 888
```

Syntax

```
VOID SetMonthDisplayOptions([ENUM FirstDayOfWeek];
                            [ENUM ShowWeekNumber];
                            [BYTE DaysToHilite])
```

Parameters

FirstDayofWeek As ENUM

(Optional) Shows which day of the week will be in the first column.

5	Friday
1	Monday
6	Saturday
0	Sunday
4	Thursday
2	Tuesday
3	Wednesday

ShowWeekNumber As ENUM

(Optional) Shows which week of the year marker at the left edge of the month control.

0 No
1 Yes

DaysToHilite As BYTE

(Optional) Changes the columns for the specified week to a red colored text. Bitwise:

0	None
1	Sun col
2	Mon col
4	Tues col
64	Sat col
65	Sat Sun cols

SetSort()

Sets the sort key and sort order for the column display.

Token ID

```
AFTKN_SET_SORT or 847
```

Syntax

```
VOID SetSort( ENUM SortKey;
              [ANSISTRING SortFieldName];
              [ENUM SortOrder])
```

Parameters

SortKey As ENUM

Specifies the sort key:

600	AcceptedCnt
112	AssignedDate
625	Author
4	Authority
3	BC
126	BeginDateTime
2	CC
6	Caller
7	Company
603	CompletedCnt
602	CopyType
139	CreateDate
626	Creator
627	CurrentVersion
176	Date
606	DeletedCnt
605	DeletedDate
507	DeliveredDate
656	DocCreatedDate
0	DocType
608	DocumentID

629	DocumentType
148	DueDate
151	EndDateTime
628	Filename
1	From
169	ItemType
613	Label
614	Library
647	LibraryId
10	Message
633	Modified
331	NamedField (for custom fields)
703	NamedFldByte
702	NamedFldDate
706	NamedFldSDWord
704	NamedFldSWord
700	NamedFldText
707	NamedFldUDWord
705	NamedFldUWord
701	NamedFldWrdStr
630	OfficialVersion
615	OpenCnt
8	Phone
5	Place
616	RecipientsCnt
617	RepliedCnt
631	RetrievedBy
632	RetrievedDate
657	SharedName
618	Size
619	Source
646	Subclass
9	Subject
646	SubType
227	TaskCategory

230	TaskPriority
351	ThreadStatus
0	To
621	TransferCnt
622	UndeliveredCnt
623	Version
634	VersionCreator
635	VersionDate
636	VersionDescription
267	ViewName

SortFieldName As ANSISTRING

Use one of the NamedFldxxx ENUMS for the SortKey. (Optional)

SortOrder As ENUM

(Optional)

111 Ascending
146 Descending

SetViewerClipboardOptions()

Displays the Clipboard Options dialog if the viewer is showing.

Token ID

```
BFTKN_CLIPBOARD_OPTIONS or 975
```

Syntax

```
VOID SetViewerClipboardOptions()
```

See Also

“SendOptions()” on page 800

SetViewerDisplayOptions()

Displays the Display Options dialog if the viewer is showing.

Token ID

```
BFTKN_DISPLAY_OPTIONS or 973
```

Syntax

```
VOID SetViewerDisplayOptions()
```

SetViewerPrintOptions()

Displays the Print Options dialog if the viewer is showing.

Token ID

```
BFTKN_PRINT_OPTIONS or 974
```

Syntax

```
VOID SetViewerPrintOptions()
```

SetWorkSchedule()

Sets the work schedule.

Token ID

```
AFTKN_SET_WORK_SCHEDULE or 890
```

Syntax

```
VOID SetWorkSchedule( [WORD StartWorkday];
                      [WORD EndWorkday];
                      [BYTE Workdays])
```

Parameters

StartWorkday As WORD

(Optional) Use minutes since midnight, so 9:00 AM is 540.

EndWorkday As WORD

(Optional) Use minutes since midnight, so 5:00 PM is 1020.

Workdays As BYTE

(Optional) Bitmask:

1	Sun
2	Mon
4	Tues
8	Wed
16	Thu
32	Fri
64	Sat
62	All but Sat and Sun

SharedFolderAccept()

Accepts a shared folder and creates a reference name and position, if necessary.

Token ID

```
AFTKN_SHARED_FOLDER_ACCEPT or 853
```

Syntax

```
VOID SharedFolderAccept( ANSISTRING FolderName;
                         ANSISTRING FolderDesc;
                         WORD FolderPosition;
                         ANSISTRING MessageID)
```

Parameters

FolderName As ANSISTRING

Requires a full folder path. For example, First Last\Cabinet\FolderName.

FolderDesc As ANSISTRING

Description of the folder.

FolderPosition As WORD

MessageID As ANSISTRING

MessageID can be obtained by using the ItemMessageIDFromView() token from the Properties view of the "Shared folder notification" message.

ShareDlg()

Shows the Sharing Tab of the Folder Properties dialog box.

Token ID

```
DTKN_FOLDER_SHARE or 65
```

Syntax

```
VOID ShareDlg( [ANSISTRING FolderName])
```

Parameters

FolderName As ANSISTRING

Full path of the folder. For example, FirstName LastName\Cabinet\FolderName. (Optional)

ShowApptAs()

Toggles the display of the folder list.

Token ID

`BFTKN_SHOWAS or 503`

Syntax

```
VOID ShowApptAs( [ENUM BusyLevel])
```

Parameters

BusyLevel As ENUM

(Optional) Specifies the busy level:

0	Free
1	Tentative
2	Busy
3	Out

ShowAttachmentWindow()

Shows or hides the Attachment Window within the QuickViewer window. The Quick Viewer Window is turned on via the View > QuickViewer menu in the client.

Token ID

```
BFTKN_SHOW_ATTWIN or 166
```

Syntax

```
VOID ShowAttachmentWindow( [ENUM State])
```

Parameters

State As ENUM

(Optional) If not specified, acts as a toggle.

0 No
1 Yes

ShowContactIndex()

Toggles the display of the contact index control. The contact index control is the list of letter buttons displayed to the right of the contact list that allow the user to jump to any point in the list like a phonebook index.

Token ID

```
BFTKN_SHOW_CONTACT_IDX or 1246
```

Syntax

```
VOID ShowContactIndex()
```

ShowFrameContents()

Shows the contents of the specified frame.

Token ID

```
BFTKN_SHOW_FRAME_CONTENTS or 1074
```

Syntax

```
ANY ShowFrameContents(ENUM WhichFrame;
                      BOOLEAN State)
```

Parameters

WhichFrame As ENUM

Indicates which frame should have its contents shown or hidden. Enumerated values:

1 FolderList
2 MessageList
3 Calendar
4 QuickViewer

State As BOOLEAN

Indicates whether to show the frame contents:

TRUE Show frame contents
FALSE Hide frame contents

Return Values

ANY

ShowFolders()

Toggles the display of the folder list.

Token ID

```
BFTKN_SHOW_FOLDERS or 408
```

Syntax

```
VOID ShowFolders()
```

ShowLongFolderList()

Displays the long folder list when QuickViewer is active.

Token ID

```
BFTKN_SHOW_LONG_FOLDERLIST 1087
```

Syntax

```
VOID ShowLongFolderList()
```

Remarks

This command acts as a toggle.

ShowMessageSourceTab()

Displays or hides the Message Source tab.

Token ID

```
BKTKN_SHOW_MESSAGESOURCE or 1183
```

Syntax

```
VOID ShowMessageSourceTab([ENUM State])
```

Parameters

State As ENUM

(Optional)

0 Off
1 On

ShowNavBar()

Displays or hides the navigation bar.

Token ID

```
BKTKN_SHOW_NAVBAR or 1221
```

Syntax

```
VOID ShowNavBar()
```

Remarks

ShowNavBar() toggles between the on and off state.

ShowQuickLook()

Toggles the display of the QuickViewer.

Token ID

```
BFTKN_SHOW_LOOKER or 407
```

Syntax

```
VOID ShowQuickLook( [ENUM State])
```

Parameters

State As ENUM

(Optional) If not specified, acts as a toggle.

0 No
1 Yes

See Also

“QuickViewerHide()” on page 705

SimpleFolderList()

Displays the simple folder list.

Token ID

`BKTKN_FOLDER_TREE_SIMPLE or 1227`

Syntax

```
VOID SimpleFolderList()
```

SortChecklistDlg()

Displays a dialog that enables the user to modify the sort criteria for the tasklist portion of an itemlist that is being viewed by tasklist.

Token ID

```
DTKN_SORT_CHECKLIST or 149
```

Syntax

```
VOID SortChecklistDlg()
```

SortDlg()

Displays the sort dialog box to sort Mailbox or Sent Items items.

Token ID

```
DTKN_SORTCOLMAN or 79
```

Syntax

```
VOID SortDlg()
```

SortFolders()

Sorts, in alphabetical order, the subfolders of a given folder.

Token ID

```
BFTKN_FOLDERS_SORT or 1033
```

Syntax

```
VOID SortFolders([ANSISTRING FolderName])
```

Parameters

FolderName As ANSISTRING

(Optional) Folder to sort children to.

Remarks

Always sorts in ascending order.

SoundAnnotationList()

Displays the Sound Annotations dialog box to play or save a sound, or get the sound annotation properties of a sound.

Token ID

```
DTKN_SOUNDS or 91
```

Syntax

```
VOID SoundAnnotationList()
```

Remarks

This token is not currently implemented in GroupWise 5.2 or newer versions.

SoundAnnotationPlay()

Plays a sound annotation.

Token ID

```
AFTKN_PLAY_SOUND or 806
```

Syntax

```
VOID SoundAnnotationPlay( WORD Index)
```

Parameters

Index As WORD

Index number (zero-based) of a sound annotation.

Remarks

This token is not currently implemented in GroupWise 5.2 or newer versions.

StatusBarGetText()

Retrieves the text in the specified pane of the main window status bar.

Token ID

```
BFTKN_STATUSBAR_GET_TEXT or 1075
```

Syntax

```
VOID StatusBarGetText([ENUM Pane])
```

Parameters

Pane As ENUM

(Optional) Enumerated values:

Number	Value	Description
1	Message	Default
2	SelectedItems	Returns the number of selected items (as a string) rather than the literal text of the pane. For example, if the pane displays "Selected: 1," the token returns "1."
3	TotalItems	Returns the total number of items (as a string) rather than the literal text of the pane. For example, if the pane displays "Total: 265," the token returns "265."

StatusBarLockText()

Locks or unlocks the main window status bar text.

Token ID

```
BFTKN_STATUSBAR_LOCK_TEXT or 1077
```

Syntax

```
BOOLEAN StatusBarLockText([ENUM Pane];
                          [BOOLEAN Lock])
```

Parameters

Pane As ENUM

(Optional) Enumerated values:

Number	Value
0	All -- Locks all three panes of the main window status bar
1	Message (default)
2	SelectedItems
3	TotalItems

Lock As BOOLEAN

(Optional) If neither value is specified, the lock state of the status bar is toggled:

TRUE Locks the specified pane of the status bar
FALSE Unlocks the specified pane of the status bar

Return Values

BOOLEAN - Previous lock state of the status bar

Remarks

Make sure the main window has focus before calling StatusBarLockText. If a message window has focus at the time StatusBarLockText is called, the function fails and the GroupWise status bar might remain locked until GroupWise is restarted.

When the status bar is locked, any status bar messages posted internally by GroupWise are not displayed on the status bar. When the status bar is locked, only messages set by StatusBarSetText() (page 848) are displayed. The token returns the previous lock state of the status bar.

StatusBarSetText()

Sets the text in the specified pane of the main window status bar.

Token ID

```
BFTKN_STATUSBAR_SET_TEXT or 1076
```

Syntax

```
VOID StatusBarSetText([ENUM Pane];
                          ANSISTRING Text)
```

Parameters

Pane As ENUM

(Optional) Enumerated values:

Number	Value	Description
1	Message	Default
2	SelectedItems	If selected, the Text parameter should be the number of selected items (as a string). If the Text parameter cannot be parsed as a valid integer, the token fails.
3	TotalItems	If selected, the Text parameter should be the total number of items (as a string). If the Text parameter cannot be parsed as a valid integer, the token fails.

Text As ANSISTRING

The text to set in the specified pane of the status bar.

Remarks

Make sure the main window has focus before calling StatusBarSetText. If a message window has focus at the time StatusBarSetText is called, the function fails.

When the status bar is locked, any status bar messages posted internally by GroupWise are not displayed on the status bar. When the status bar is locked, only messages set by this function are displayed. The token returns the previous lock state of the status bar.

StatusWindowShow()

Show or hidess the status window that gives feedback while connecting to the master in remote or while connecting to pop and imap servers.

Token ID

```
BFTKN_SHOW_STATUS_WINDOW or 1002
```

Syntax

```
VOID StatusWindowShow()
```

Remarks

This token is a toggle. If the Status Window is visible, it hides the window. If the Status Window is hidden, the token shows the window.

SwitchToHTMLView()

Switches the editor to use HTML if it isn't in this mode already. Valid on compose mail view.

Token ID

```
BFTKN_VIEW_HTML or 990
```

Syntax

```
VOID SwitchToHTMLView()
```

SwitchToRTFView()

Switches the editor to use the RTF (plain text) mode if it is not in this mode already.

Token ID

```
BFTKN_VIEW_RTF or 989
```

Syntax

```
VOID SwitchToRTFView()
```

T-U

This section contains information about the following tokens:

TabNext()

Moves the insertion point forward a specific number of tab stops.

Token ID

```
BFTKN_VIEW_TAB or 254
```

Syntax

```
VOID TabNext( [WORD Count])
```

Parameters

Count As WORD

Number of tab stops. Default: 1. (Optional)

See Also

“FocusSet()” on page 357

TabPrevious()

Moves the insertion point backwards a specified number of tab stops. Not recordable.

Token ID

```
BFTKN_VIEW_SHIFTTAB or 255
```

Syntax

```
VOID TabPrevious( [WORD Count])
```

Parameters

Count As WORD

Definition (Domain.PostOffice.UserID). (Optional)

TextSetAuthority()

Inserts the name of a person or entity in an Authority text box (available only in customized Appointment view).

Token ID

```
AFTKN_SET_AUTHORITYTEXT or 573
```

Syntax

```
VOID TextSetAuthority( ANSISTRING AuthorityText;
                       [BOOLEAN Append])
```

Parameters

AuthorityText As ANSISTRING

Append As BOOLEAN

(Optional) Specifies the following:

False Default. Replaces existing text.
True Appends to existing text.

TextSetABEntry()

Purpose Here.

Token ID

```
AFTKN_SET_ABENTRY or 601
```

Syntax

```
VOID TextSetABEntry( ANSISTRING ABEntry;
                     ENUM EntryType;
                     [BOOLEAN Append])
```

Parameters

ABEntry As ANSISTRING

Definition (Domain.PostOffice.UserID)

EntryType As ENUM

Specifies the entry type:

3	BC
2	CC
0	To

Append As BOOLEAN

Definition (Domain.PostOffice.UserID) (Optional)

TextSetBC()

Inserts a user ID in a BC text box.

Token ID

```
AFTKN_SET_BCTEXT or 569
```

Syntax

```
VOID TextSetBC( ANSISTRING BCText;
                [BOOLEAN Append])
```

Parameters

BCText As ANSISTRING

User ID of blind copy recipients. Separate multiple users with commas.

Append As BOOLEAN

(Optional) Specifies the following:

False Default. Replaces existing text.
True Appends to existing text.

TextSetCallerName()

Inserts a caller's name in the Caller text box of a Phone Message view.

Token ID

```
AFTKN_SET_NAMETEXT or 575
```

Syntax

```
VOID TextSetCallerName( ANSISTRING CallerName;
                        [BOOLEAN Append])
```

Parameters

CallerName As ANSISTRING

Name of the person who called.

Append As BOOLEAN

(Optional) Specifies the following:

False Default. Replaces existing text.
True Appends to existing text.

TextSetCategoryAndPriority()

Inserts a character in the Priority text box of a Task view.

Token ID

```
AFTKN_SET_CAT_PRIOR or 590
```

Syntax

```
VOID TextSetCategoryAndPriority( ANSISTRING CatAndPrior;
                                 [BOOLEAN Append])
```

Parameters

CatAndPrior As ANSISTRING

For example: "A5" (A is the category and 5 the priority.

Append As BOOLEAN

(Optional) Specifies the following:

False Default. Replaces existing text.
True Appends to existing text.

TextSetCC()

Inserts a user ID in a CC text box.

Token ID

```
AFTKN_SET_CCTEXT or 568
```

Syntax

```
VOID TextSetCC( ANSISTRING CCText;
                [BOOLEAN Append])
```

Parameters

CCText As ANSISTRING

User ID of carbon copy recipients. Separate multiple users with commas.

Append As BOOLEAN

(Optional) Specifies the following:

False Default. Replaces existing text.
True Appends to existing text.

TextSetCompany()

Inserts a company’s name in the Company text box of a Phone view.

Token ID

```
AFTKN_SET_COMPANYTEXT or 576
```

Syntax

```
VOID TextSetCompany( ANSISTRING CompanyText;
                     [BOOLEAN Append])
```

Parameters

CompanyText As ANSISTRING

Name of the company.

Append As BOOLEAN

(Optional) Specifies the following:

False Default. Replaces existing text.
True Appends to existing text.

TextSetDateTime()

Inserts a date and time in the Appointment date and time text boxes of an Appointment view.

Token ID

```
AFTKN_SET_DATETIMEDUR or 584
```

Syntax

```
TextSetDateTime( [WORD StartDay];
                 [WORD StartMonth];
                 [WORD StartYear];
                 [WORD StartMinute];
                 [WORD StartHour];
                 [WORD EndDay];
                 [WORD EndMonth];
                 [WORD EndYear];
                 [WORD EndMinute];
                 [WORD EndHour];
                 [WORD DurationMinutes];
                 [WORD DurationHours])
```

Parameters

StartDay As WORD

(Optional) Day when the appointment starts.

StartMonth As WORD

(Optional) Month when the appointment starts.

StartYear As WORD

(Optional) Year when the appointment starts.

StartMinute As WORD

(Optional) Minute when the appointment starts.

StartHour As WORD

Military time (24-hour clock). (Optional)

EndDay As WORD

(Optional) Day when the appointment ends.

EndMonth As WORD

(Optional) Month when the appointment ends.

EndYear As WORD

(Optional) Year when the appointment ends.

EndMinute As WORD

(Optional) Minute when the appointment ends.

EndHour As WORD

(Optional) Military time (24-hour clock).

DurationMinutes As WORD

(Optional) Number of minutes that the appointment lasts (in addition to the number of hours).

DurationHours As WORD

(Optional) Number of hours that the appointment lasts.

Remarks

If Date Time Options, Calendar Tab, Appointment Options, or Display Event Length is set to Duration, do not use End parameters. If Date Time Options, Calendar Tab, Appointment Options, or Display Event Length is set to End Date and End Time, do not use Duration parameters. (See “PrefAppointmentTime()” on page 648.)

TextSetFrom()

Inserts a user's name in the From text box. The logged user name is included in parenthesis.

Token ID

```
AFTKN_SET_FROMTEXT or 567
```

Syntax

```
VOID TextSetFrom( ANSISTRING FromText;
                  [BOOLEAN Append])
```

Parameters

FromText As ANSISTRING

Name of the user that the item is from.

Append As BOOLEAN

(Optional) Specifies the following:

False Default. Replaces existing text.
True Appends to existing text.

TextSetMessage()

Inserts the text contained in MessageText into the Message field of the currently active compose view. The text is either appended to the message, or replaces the text of the message, based on the value of Append.

Token ID

```
AFTKN_SET_MSGTEXT or 571
```

Syntax

```
VOID TextSetMessage( ANSISTRING MessageText;
                     [BOOLEAN Append])
```

Parameters

MessageText As ANSISTRING

Text to insert into the message.

Append As BOOLEAN

(Optional) Specifies the following:

False Default. Replaces existing text.
True Appends to existing text.

Remarks

The message body field (in the Message View and the Object API) has a limit of 32,000 characters. The Message View will stop accepting data after 32,000 characters.

TextSetPhoneNumber()

Inserts text in the Phone text box of a Phone Message view.

Token ID

```
AFTKN_SET_PHONETEXT or 577
```

Syntax

```
VOID TextSetPhoneNumber( ANSISTRING PhoneNumber;
                         [BOOLEAN Append])
```

Parameters

PhoneNumber As ANSISTRING

Phone number associated with the phone message.

Append As BOOLEAN

(Optional) Specifies the following:

False Default. Replaces existing text.
True Appends to existing text.

TextSetPlaceName()

Inserts a location in the Place text box of an Appointment view.

Token ID

```
AFTKN_SET_PLACETEXT or 574
```

Syntax

```
VOID TextSetPlaceName( ANSISTRING PlaceName;
                       [BOOLEAN Append])
```

Parameters

PlaceName As ANSISTRING

Name of the place for the appointment.

Append As BOOLEAN

(Optional) Specifies the following:

False Default. Replaces existing text.
True Appends to existing text.

TextSetSubject()

Inserts a subject in a Subject text box.

Token ID

```
AFTKN_SET_SUBJTEXT or 570
```

Syntax

```
VOID TextSetSubject( ANSISTRING SubjectText;
                     [BOOLEAN Append])
```

Parameters

SubjectText As ANSISTRING

Text for the subject of the item

Append As BOOLEAN

(Optional) Specifies the following:

False Default. Replaces existing text.
True Appends to existing text.

TextSetTo()

Inserts a user ID in a To text box.

Token ID

```
AFTKN_SET_TOTEXT or 566
```

Syntax

```
VOID TextSetTo( ANSISTRING ToText;
                [BOOLEAN Append])
```

Parameters

ToText As ANSISTRING

Name of the user who the item is being sent to.

Append As BOOLEAN

(Optional) Specifies the following:

False Default. Replaces existing text.
True Appends to existing text.

Thesaurus()

Displays the Thesaurus dialog box.

Token ID

```
BFTKN_THES or 253
```

Syntax

```
VOID Thesaurus()
```

TimeSetEndTimeMode()

Displays the Time Input dialog box.

Token ID

```
DTKN_GSET_ENDTIME or 77
```

Syntax

```
VOID TimeSetEndTimeMode()
```

TimeSetStartTimeMode()

Displays the Time Input dialog box.

Token ID

```
DTKN_GSET_STARTTIME or 76
```

Syntax

```
VOID TimeSetStartTimeMode()
```

TimeZoneLabel()

Set a label for a timezone that appears on the calendar.

Token ID

```
AFTKN_TIMEZONE_LABEL or 934
```

Syntax

```
VOID TimeZoneLabel(
     [ANSISTRING] TimeZoneRegKey;
     [ANSISTRING] TimeZoneLabel)
```

Parameters

TimeZoneRegKey As ANSISTRING

Identifies one of the timezone subkeys under “HKEY_LOCAL_MACHINE\Software\Microsoft\Windows NT\Time Zones”. For instance, “Alaskan Standard Time”.

TimeZoneLabel As ANSISTRING

The new label for the timezone.

Type()

Inserts text into a text box at the insertion point.

Token ID

```
AFTKN_TYPE or 602
```

Syntax

```
VOID Type( ANSISTRING Text)
```

Parameters

Text As ANSISTRING

Text to insert into the text box.

Remarks

The message body field (in the Message View and the Object API) has a limit of 32,000 characters. The Message View will stop accepting data after 32,000 characters.

TypeChar()

Inserts a character at the insertion point.

Token ID

```
AFTKN_TYPE_CHAR or 603
```

Syntax

```
VOID TypeChar( WORD CharSet; WORD CharNum)
```

Parameters

CharSet As WORD

Definition (Domain.PostOffice.UserID)

CharNum As WORD

Definition (Domain.PostOffice.UserID)

UpLevel()

Selects the attachment that is up one level.

Token ID

```
BFTKN_UP_LEVEL or 203
```

Syntax

```
VOID UpLevel()
```

See Also

“Type()” on page 874

UserFunction()

Executes a command from another application. Primarily used by third-party developers.

Token ID

```
AFTKN_USER_FUNCTION or 610
```

Syntax

```
VOID UserFunction( ANSISTRING Command)
```

Parameters

Command As ANSISTRING

Command to execute a third-party DLL. The command name should take the form xxxx: name, where xxxx represents a four-character identifier.

V-Z

This section contains information about the following tokens:

VacationRuleDlg()

Display a dialog enabling the user to create or modify a vacation rule.

Token ID

```
DTKN_VACATION_RULE_DLG or 151
```

Syntax

```
VOID VacationRuleDlg()
```

ViewContacts()

Displays or hides contacts in the contact list.

Token ID

```
BFTKN_PEOPLE_FLT 1095
```

Syntax

```
VOID ViewContacts([ENUM State])
```

Parameters

State as ENUM

(Optional) Specifies the following:

0 Off Filters out contacts, hiding them
1 On Shows contacts in the list

Remarks

If the State parameter is not passed in, this command acts as a toggle.

ViewContactsOnly()

Displays only the contacts in the contact list (and filters out groups, references, and organizations).

Token ID

```
BFTKN_PEOPLE_FLT_ONLY 1114
```

Syntax

```
VOID ViewContactsOnly()
```

ViewDocument()

Allows the document specified by the Document Number in DocIDStr to be displayed in the viewer.

Token ID

```
AFTKN_DM_VIEW_DOC or 859
```

Syntax

```
VOID ViewDocument( ANSISTRING DocIDStr)
```

Parameters

DocIDStr As ANSISTRING

Document number of the document to display.

ViewDraftItems()

Toggles the display of draft items in the GroupWise client, based on the value of State.

Token ID

```
BFTKN_DRAFT_FLT or 410
```

Syntax

```
VOID ViewDraftItems( [ENUM State])
```

Parameters

State As ENUM

If not specified, acts as a toggle.

0 Off!
1 On!

ViewDraftItemsOnly()

Allows the GroupWise client to display only draft items.

Token ID

`BFTKN_DRAFT_FLT_ONLY or 427`

Syntax

```
VOID ViewDraftItemsOnly()
```

ViewHideCompletedItemsImmediately()

Toggles a setting that controls whether or not tasklist items are hidden as soon as they are completed.

Token ID

```
BFTKN_CHECKLIST_HIDE_COMPLETED_FLT or 1119
```

Syntax

```
VOID ViewHideCompletedItemsImmediately([ENUM] State)
```

Parameters

State As ENUM

(Optional) One of the following values

0 Off
1 On

ViewHideCompletedItemsNextDayAfterCompleted()

Toggles a setting that controls whether or not tasklist items are hidden on the day following their completion.

Token ID

`BFTKN_CHECKLIST_HIDE_COMPLETED_AFTER_A_DAY_FLT or 1120`

Syntax

```
VOID ViewHideCompletedItemsNextDayAfterCompleted([ENUM] State)
```

Parameters

State As ENUM

(Optional) One of the following values

0 Off
1 On

ViewerPageDown()

Pages the viewer window down.

Token ID

```
BFTKN_VIEWER_PAGEDOWN or 492
```

Syntax

```
VOID ViewerPageDown()
```

ViewerPageUp()

Pages the viewer window up.

Token ID

```
BFTKN_VIEWER_PAGEUP or 491
```

Syntax

```
VOID ViewerPageUp()
```

ViewGroups()

Displays or hides groups in the contact list.

Token ID

```
BFTKN_GROUP_FLT 1096
```

Syntax

```
VOID ViewGroups([ENUM State])
```

Parameters

State as ENUM

(Optional) Specifies the following:

0 Off Filters out groups, hiding them
1 On Shows groups in the list

Remarks

If the State parameter is not passed in, this command acts as a toggle.

ViewGroupsOnly()

Displays only the groups in the contact list (and filters out contacts, references, and organizations).

Token ID

```
BFTKN_GROUP_FLT_ONLY 1115
```

Syntax

```
VOID ViewGroupsOnly()
```

ViewHideCaption()

Shows or hides the title bar and menu for the view.

Token ID

```
BKTKN_HIDE_CAPTION or 197
```

Syntax

```
VOID ViewHideCaption([ENUM State])
```

Parameters

State As ENUM

(Optional)

281 Hide
282 Show

ViewHideMenu()

Shows or hides the menu of the view.

Token ID

```
BKTKN_HIDE_MENU or 1219
```

Syntax

```
VOID ViewHideMenu([ENUM State])
```

Parameters

State As ENUM

(Optional)

281 Hide
282 Show

ViewHideNonChecklistItems()

Displays or hides non checklist items in the contact list.

Token ID

```
BFTKN_CHECKLIST_HIDE_NON_CHECKLIST_FLT 1094
```

Syntax

```
VOID ViewHideNonChecklistItems([ENUM State])
```

Parameters

State as ENUM

(Optional) Specifies the following:

0 Off
1 On

ViewItem()

Brings up the viewer for the currently selected item and its attachments.

Token ID

```
BFTKN_VIEW_ITEM or 409
```

Syntax

```
VOID ViewItem()
```

ViewMaximize()

Enlarges a window or view to full size. Items reduced to an icon can only be restored (see ViewRestore()), they cannot be maximized.

Token ID

```
BFTKN_MAXIMIZE_WINDOW or 161
```

Syntax

```
VOID ViewMaximize()
```

See Also

“ViewMinimize()” on page 896
“ViewRestore()” on page 913

ViewMinimize()

Reduces a window or view to an icon.

Token ID

```
BFTKN_MINIMIZE_WINDOW or 268
```

Syntax

```
VOID ViewMinimize()
```

See Also

“ViewMaximize()” on page 895
“ViewRestore()” on page 913

ViewNextAttach()

Views the next item in the list.

Token ID

```
BFTKN_VIEW_NEXT_ATTACH or 495
```

Syntax

```
VOID ViewNextAttach()
```

ViewOnTop()

Displays Mailbox, Sent Items, Trash, or Calendar views and icons on top of other views.

Token ID

```
BFTKN_ON_TOP or 198
```

Syntax

```
VOID ViewOnTop( [ENUM OnTop])
```

Parameters

OnTop As ENUM

(Optional) If no parameter is specified, acts like a toggle.

0 No
1 Yes

See Also

"ViewMinimize()" on page 896

ViewOpen()

Opens a default view such as a Mail or Calendar view.

Token ID

```
AFTKN_OPEN_VIEW_TYPE or 554
```

Syntax

```
VOID ViewOpen( ENUM ViewType;
               [ENUM IsGroupItem];
               [ANSISTRING FolderName];
               [DWORD ItemEngineFolder])
```

Parameters

ViewType As ENUM

Specifies the view type:

274	Appointment
624	Browser
1	Calendar
658	Discussion
350	DiscussionNNTP
6	Inbox
276	Mail
278	Note
2	OldAppointment
0	OldMail
4	OldNote
5	OldPhone
3	OldTask
7	Outbox
288	PhoneMessage
277	Task
8	Trash

IsGroupItem As ENUM

(Optional) Refers to a personal/posted item viewtype as opposed to an item message with recipients. Enumerated values:

0 personal/posted

1 with recipients (default)

FolderName As ANSISTRING

(Optional) FolderName is used with ViewType ENUM of Inbox 6 to select a specified folder in your folder list. For example, ViewOpen(6;;"FirstName LastName\Cabinet") selects the Cabinet folder.

ItemItemEngineFolder As DWORD

(Optional)

See Also

“NewAppointment()” on page 572
“NewMail()” on page 576
“NewNote()” on page 578
“NewPhone()” on page 581
“NewTask()” on page 585
“OpenCalendar()” on page 617
“OpenInBox()” on page 620
“OpenOutBox()” on page 622
“OpenTrashWindow()” on page 623
“PrefViewDefaults()” on page 667
“ViewOpenFile()” on page 902
“ViewOpenNamed()” on page 903

ViewOpenDlg()

Displays the Open View dialog box.

Token ID

```
DTKN_FILE_OPEN_DLG or 32
```

Syntax

```
VOID ViewOpenDlg()
```

See Also

“ViewOpen()” on page 899

ViewOpenFile()

Opens a view by its file name.

Token ID

```
AFTKN_OPEN_VIEW_FILE or 588
```

Syntax

```
VOID ViewOpenFile( ANSISTRING FileName;
                   [ANSISTRING FolderName];
                   [ANSISTRING ClassName])
```

Parameters

Filename As ANSISTRING

Include full path.

FolderName As ANSISTRING

FolderName allows you to specify where any Personal/Posted message item will stored. Not valid for group type messages. (Optional)

ClassName As ANSISTRING

Added to the end of a message class. For example, GW.MESSAGE.MAIL.ClassName. (Optional)

See Also

“ViewOpen()” on page 899

ViewOpenNamed()

Opens a view by menu name and type. Some valid shipping view names for message items are: Mail, Mail (small), Meeting, Task, Reminder, and Phone Message. For calendar views: Day, Week, Month, Year, etc.

Token ID

```
AFTKN_OPEN_VIEW_NAMED or 587
```

Syntax

```
VOID ViewOpenNamed( ANSISTRING ViewName;
                    ENUM ViewType;
                    [ANSISTRING FolderPath])
```

Parameters

ViewName As ANSISTRING

Example: "Mail" or "Appointment w/attach"

ViewType As ENUM

Specifies the view type:

274	Appointment
1	Calendar
276	Mail
278	Note
288	PhoneMessage
277	Task

FolderPath As ANSISTRING

(Optional)

See Also

“ViewOpen()” on page 899
“ViewOpenFile()” on page 902

ViewOrganizations()

Displays or hides organizations in the contact list.

Token ID

```
BFTKN_ORGANIZATION_FLT 1098
```

Syntax

```
VOID ViewOrganizations([ENUM State])
```

Parameters

State as ENUM

(Optional) Specifies the following:

0 Off Filters out organizations, hiding them
1 On Shows organizations in the list

Remarks

If the State parameter is not passed in, this command acts as a toggle.

ViewOrganizationsOnly()

Displays only the organizations in the contact list (and filters out contacts, groups, and references).

Token ID

```
BFTKN_ORGANIZATION_FLT_ONLY 1117
```

Syntax

```
VOID ViewOrganizationsOnly()
```

ViewPersonalItems()

Toggles the display of personal or posted items in the GroupWise client, based on the value of State.

Token ID

`BFTKN_PERS_FLT or 384`

Syntax

```
VOID ViewPersonalItems( [ENUM State])
```

Parameters

State As ENUM

(Optional) If not specified, acts as a toggle.

0 Off!
1 On!

ViewPersonalItemsOnly()

Allows the GroupWise client to only display personal or posted items.

Token ID

`BFTKN_PERS_FLT_ONLY or 426`

Syntax

```
VOID ViewPersonalItemsOnly()
```

ViewPrevAttach()

Views the previous item in the list.

Token ID

```
BFTKN_VIEW_PREV_ATTACH or 496
```

Syntax

```
VOID ViewPrevAttach()
```

ViewReceivedItems()

Toggles the display of received items in the GroupWise client based on the value of State.

Token ID

`BFTKN_INBOX_FLT or 380`

Syntax

```
VOID ViewReceivedItems( [ENUM State])
```

Parameters

State As ENUM

(Optional) If not specified, acts as a toggle.

0 Off!
1 On!

ViewReceivedItemsOnly()

Allows the GroupWise client to only display received items.

Token ID

```
BFTKN_INBOX_FLT_ONLY 424
```

Syntax

```
VOID ViewReceivedItemsOnly()
```

ViewResources()

Displays or hides resources in the contact list.

Token ID

```
BFTKN_RESOURCE_FLT 1097
```

Syntax

```
VOID ViewResources([ENUM State])
```

Parameters

State as ENUM

(Optional) Specifies the following:

0 Off Filters out resources, hiding them
1 On Shows resources in the list

Remarks

If the State parameter is not passed in, this command acts as a toggle.

ViewResourcesOnly()

Displays only the resources in the contact list (and filters out contacts, groups, and organizations).

Token ID

```
BFTKN_RESOURCE_FLT_ONLY 1116
```

Syntax

```
VOID ViewResourcesOnly()
```

ViewRestore()

Restores a window to its previous size and position.

Token ID

```
BFTKN_RESTORE_WINDOW or 269
```

Syntax

```
VOID ViewRestore()
```

ViewSentItems()

Toggles the display of sent items in the GroupWise client, based on the value of State.

Token ID

```
BFTKN_OUTBOX_FLT 383
```

Syntax

```
VOID ViewSentItems( [ENUM State])
```

Parameters

State As ENUM

If not specified, acts as a toggle. (Optional)

0 Off!
1 On!

ViewSentItemsOnly()

Allows the GroupWise client to only display sent items.

Token ID

```
BFTKN_OUTBOX_FLT_ONLY 425
```

Syntax

```
VOID ViewSentItemsOnly()
```

ViewSource()

View the source of an RTF (non-HTML) message.

Token ID

```
BFTKN_VIEW_SOURCE or 1243
```

Syntax

```
VOID ViewSource()
```

ViewSwitch()

Opens a view from a view, copies the information from the old view to the new, and then closes the old view. The ViewName parameter is required if the ViewType parameter is used, and vice versa.

Token ID

```
AFTKN_CHANGE_VIEW or 629
```

Syntax

```
VOID ViewSwitch( [ANSISTRING Filename];
                 [ENUM ViewType];
                 [ANSISTRING ViewName])
```

Parameters

Filename As ANSISTRING

Open a view by file name. With this parameter, do not use ViewType and ViewName. (Optional)

ViewType As ENUM

Open a view by menu name and type. With this parameter, do not use Filename. (Optional)

274	Appointment
1	Calendar
276	Mail
278	Note
7	Outbox
288	PhoneMessage
277	Task

ViewName As ANSISTRING

Menu name such as "Mail" or "Appointment w/attach" (see the ViewType parameter). (Optional)

ViewSwitchDlg()

Displays the Switch View dialog box.

Token ID

```
DTKN_CHANGE_VIEW or 86
```

Syntax

```
VOID ViewSwitchDlg()
```

ViewVersionList()

Displays the version list of the document specified by DocNum in the library specified by Library. WhereToSearch determines whether the version list is obtained from the Master or Remote post office.

Token ID

```
AFTKN_DM_VERSION_LIST or 844
```

Syntax

```
VOID ViewVersionList( DWORD DocNum;
                      [ENUM WhereToSearch];
                      [ANSISTRING Libraries])
```

Parameters

DocNum As DWORD

The document number of the document to display a version list for.

WhereToSearch As ENUM

(Optional) Specifies the following:

4 MasterMailBox!
2 RemoteMailBox!

Libraries As ANSISTRING

(Optional)

WritingToolsTextLanguage()

Displays the Writing Tools Text Language dialog box.

Token ID

```
BFTKN_WT_TEXT_LANGUAGE or 429
```

Syntax

```
VOID WritingToolsTextLanguage()
```

7 Structures

This section contains the structures, types, and enumerations for GroupWise Tokens in the following sections:

MAC_PARAM

A structure used to store token parameter information.

Definition

```
typedef struct _tagMAC_PARAM
{
   MAC_VALUE_TYPE eType;
   WORD wFlags;
   WORD wReserved;
   union
   {
      DWORD dwValue;
      void far *lpvPtr;
      double drValue;
   } uData;
} MAC_PARAM, NEAR *NPMAC_PARAM, FAR *LPMAC_PARAM,
   *PMAC_PARAM;
```

Types

The following types are used:

MAC_PARAM	Structure containing token parameter information.
NPMAC_PARAM	Near pointer to a MAC_PARAM structure.
LPMAC_PARAM	Far pointer to a MAC_PARAM structure.
PMAC_PARAM	Pointer to a MAC_PARAM structure.

Members

The members are defined as follows:

eType	Parameter type enumeration. See “MAC_VALUE_TYPE” on page 928.
wFlags	When using the token parameter, set wFlags to 00000. When the token parameter is optional and unused, set wFlags to 08000.
uData	Union containing parameter data.
	dwValue = Integer data
	lpvPtr = String data
	drValue = Floating point data

MAC_RETURNVAL

Used to hold the return value information for value-returning tokens.

Definition

```
typedef struct _tagMAC_RETURNVAL
{
   HSZ             hszRequestor;
   MAC_MACROID     dwMacroID;
   MAC_TOKENERROR  eReturnCode;
   MAC_VARIABLE    rv;
}
MAC_RETURNVAL, NEAR *NPMAC_RETURNVAL,
   FAR *LPMAC_RETURNVAL, *PMAC_RETURNVAL;

typedef LPMAC_RETURNVAL FAR *LPLPMAC_RETURNVAL;
```

Types

The following types are used:

MAC_RETURNVAL	Structure holding the token return value information.
NPMAC_RETURNVAL	Near pointer to a MAC_RETURNVAL structure.
LPMAC_RETURNVAL	Far pointer to a MAC_RETURNVAL structure.
LPLPMAC_RETURNVAL	Far pointer to a LPMAC_RETURNVAL.

Members

The members are defined as follows:

hszRequestor	Unique application identifier that sent the token.
dwMacroID	Token application identifier. Should always be 0L.
eReturnCode	Token error return code. See “MAC_TOKENERROR” on page 926.
rv	Token return value. See “MAC_VARIABLE” on page 931

MAC_TOKEN

Structure used to represent GroupWise tokens. It contains information identifying the token as well as token parameter data.

When allocating memory for a MAC_TOKEN structure, memory for the first parameter is provided. If the token has more than one parameter, additional contiguous memory must be allocated for MAC_PARAM.

Definition

```
typedef struct _tagMAC_TOKEN
{
   HSZ              hszCommand;
   MAC_IPCVERSION   Version;
   HSZ              hszRequestor;
   MAC_MACROID      dwMacroID;
   ATOM             atomApp;
   WORD             wReserved;
   MAC_TOKENID      wTokenId
   MAC_COUNT        cParam;
   DWORD            dwFlags;
   DWORD            dwReserved;
   MAC_PARAM        rgParam[1];
} MAC_TOKEN, NEAR *NPMAC_TOKEN, FAR *LPMAC_TOKEN, *PMAC_TOKEN;
```

Types

The following types are used:

MAC_TOKEN	Structure used to represent actual tokens.
NPMAC_TOKEN	Near pointer to a MAC_TOKEN structure.
LPMAC_TOKEN	Far pointer to a MAC_TOKEN structure.
PMAC_TOKEN	Pointer to a MAC_TOKEN structure.

Members

The members are defined as follows:

hszCommand	Indicates where the token originated.
Version	Indicates the IPC version and must be set to 0 (==0).
hszRequestor	Unique application identifier.
dwMacroID	Application ID for this macro token. Should always be 0L.
atomApp	Where the token is bound.
wTokenId	Token ID
cParam	Number of token parameters.
dwFlags	Writeable or non-writeable.
dwReserved	Reserved.
rgParam	Parameter data block pointer. rgParam[0] contains the data for the first parameter. Make sure you allocate memory next to the first parameter for added parameters. See “MAC_PARAM” on page 922.

MAC_TOKENERROR

An enumeration list used to define the various error conditions resulting from token processing.

Definition

```
typedef enum
{
   MAC_FUNCTION_OK = MAC_NO_ERROR,
   MAC_FUNCTION_UNKNOWN = MAC_BASE_GENERAL_ERRORS,
   MAC_FUNCTION_NOTFOUND,
   MAC_FUNCTION_CANCEL,
   MAC_FUNCTION_ERROR,
   MAC_FUNCTION_INVALID_PARM,
   MAC_FUNCTION_INVALID,
   MAC_FUNCTION_NOT_HANDLED,
   MAC_FUNCTION_RETURN_LATER
} MAC_TOKENERROR;
```

Types

The following types are used:

MAC_TOKENERROR	Enumerated token processing error type.

Members

The members are defined as follows:

MAC_FUNCTION_OK	Function returned with no error condition.
MAC_NO_ERROR	Function returned with no error condition.
MAC_FUNCTION_UNKNOWN	Function is unknown.
MAC_BASE_GENERAL_ERRORS	Function is unknown.
MAC_FUNCTION_NOTFOUND	Function resulted in a Not Found condition.
MAC_FUNCTION_CANCEL	Function resulted in a Cancel condition.
MAC_FUNCTION_ERROR	Function resulted in an Error condition.
MAC_FUNCTION_INVALID_PARM	Function contained an invalid parameter.
MAC_FUNCTION_INVALID	Function is invalid at this time.
MAC_FUNCTION_NOT_HANDLED	Function was not handled.
MAC_FUNCTION_RETURN_LATER	Delays from token handling function returns.

MAC_TOKENID

A defined type used to pass token values

Definition

```
typedef WORD              MAC_FUNCTIONID
typedef MAC_FUNCTIONID    MAC_TOKENID
```

MAC_VALUE_TYPE

An enumerated type which defines the various parameter types supported by tokens.

Definition

```
typedef enum
{
   eValUndefined =  VALUE_GENERAL_TYPES,
   eParmAny = VALUE_PARAMETER_TYPES,
   eParmBoolean,
   eParmByteSigned,
   eParmByteUnsigned,
   eParmCentimeters,
   eParmDwordSigned,
   eParmDwordUnsigned,
   eParmDWPUSigned,
   eParmDWPUUnsigned,
   eParmEnumeration,
   eParmFloat,
   eParmInches,
   eParmMillimeters,
   eParmPoints,
   eParmStringAnsi,
   eParmStringLabel,
   eParmStringOem,
   eParmStringWord,
   eParmStringVariable,
   eParmTokenID,
   eParmWordSigned,
   eParmWordUnsigned,
   eParmWPFname,
   eParmWPUSigned,
   eParmWPUUnsigned,
   eParmUserDialog,
   eParmStringWide,
   eValArrayDefinition = VALUE_VARIABLE_TYPES,
   eValBoolean,
   eValCentimeters,
   eValFloat,
   eValInches,
   eValInteger,
   eValMillimeters,
   eValOLEObject,
   eValPoints,
   eValString,
   eValStringAnsi,
   eValStringOem,
   eValWPUnits,
   eValAlias,
   eValRecord,
   eValDateTime,
   eValRawBinary,
   eValZzzzzNoMore = VALUE_END_OF_TYPES
} MAC_VALUE_TYPE;
```

Types

The following types are used:

MAC_VALUE_TYPE	Enumeration defining the various token parameter types.

Members

The members are defined as follows:

eParmUndefined	Undefined parameter type. Should not be used in third-party applications.
eParmAny	Parameter type decided at run time. Should not be used in third-party applications.
eParmBoolean	C Boolean type.
eParmByteSigned	Signed 8-bit value.
eParmByteUnsigned	Unsigned 8-bit value.
eParmCentimeters	Double.
eParmDwordSigned	Signed 32-bit value.
eParmDwordUnsigned	Unsigned 32-bit value.
eParmDWPUSigned	Signed 32-bit unit (WPU).
eParmDWPUUnsigned	Unsigned 32-bit unit (WPU)
eParmEnumeration	Enumeration 16-bit value.
eParmFloat	Double
eParmInches	Double
eParmMillimeters	Double
eParmPoints	Double
eParmStringAnsi	Null-terminated ANSI string.
eParmStringLabel	Null-terminated WPC word string containing a label reference.
eParmStringOem	Null-terminated OEM string.
eParmStringWord	Null-terminated WPC word string.
eParmStringVariable	Word string containing a variable name.
eParmTokenID	Unsigned 16-bit value.
eParmWordSigned	Signed 16-bit value.
eParmWordUnsigned	Unsigned 16-bit value.
eParmWPFname	Null-terminated ANSI string containing a file name.
eParmWPUSigned	Signed 16-bit unit (WPU).
eParmWPUUnsigned	Unsigned 16-bit unit (WPU).
eParmUserDialog	User dialog buffer (like a string).

eValArrayDefinition	Array definition variable type.
eValBoolean	C Boolean variable type.
eValCentimeters	Double variable type.
eValFloat	Double variable type.
eValInches	Double variable type.
eValInteger	Signed 32-bit variable type.
eValMillimeters	Double variable type.
eValPoints	Double variable type.
eValString	Null-terminated Word String variable type.
eValStringAnsi	Null-terminated ANSI string variable type.
eValStringOem	Null-terminated OEM string variable type.
eValWPUnits	Signed 32-bit variable type.
eValZzzzzNoMore	Not used. Only an indication of final value in enumeration.

MAC_VARIABLE

Structure containing the return value data for value-returning tokens.

Definition

```
typedef struct _tagMAC_VARIABLE
{
HSZ             hszCommand;
MAC_IPCVERSION  Version;
HSZ             hszRequestor;
MAC_MACROID     dwMacroID;
ATOM            atomApp;
WORD            wReserved;
MAC_VALUE_TYPE  eType;
MAC_TOKENID     wSysVarID;
MAC_SYMBOLNAME  wzSymbolName
union
{
   BOOL          fData;
   VALUE_FLOAT   drData;
   VALUE_INT     lData;
   struct
   {
      WORD       wStrLen;
      union
         {
            WZ       wzData[1];
            AZ       azData[1];
         } uStr;
      } stData;
   } uValue;
} MAC_VARIABLE, *PMAC_VARIABLE, FAR *LPMAC_VARIABLE;
```

Types

The following types are used:

MAC_VARIABLE	Structure containing the return value information for a token.
LPMAC_VARIABLE	Far pointer to a MAC_VARIABLE structure.
MAC_IPCVERSION	
PMAC_VARIABLE	Pointer to a MAC_VARIABLE structure.

Members

The members are defined as follows:

hszCommand	Structure containing the return value information for a token.
Version	Indicates the IPC version and must be set to 0 (==0).
hszRequestor	Far pointer to a MAC_VARIABLE structure.
dwMacroID	
atomApp	Indicates the IPC version and must be set to 0 (==0).
wSysVarID	System variable identifier.
wzSymbolName	Word string containing the symbol name.
eType	Return value type. Must be in the range between VALUE_VARIABLE_TYPES+1 and VALUE_END_OF_TYPES. See “MAC_VALUE_TYPE” on page 928.
uValue	Union containing the actual return value.
	fData Boolean data.drData Floating point data.lData Integer data.stData String data union.wStrLen Character length of string data.uStr String union.wzData Word data.azData ANSI string data.

TPH_RETURNVAL

Structure used to pass token information to HandleTokenValReturn( ).

Definition

```
typedef struct tph_returnval_tag
{
   LPMAC_TOKEN lpToken;
   LPLPMAC_RETURNVAL lplpmacRetVal;
} TPH_RETURNVAL, FAR *LPTPH_RETURNVAL;
```

Types

The following types are used:

TPH_RETURNVAL	The third-party handler return value structure.
LPTPH_RETURNVAL	Far pointer to a TPH_RETURNVAL structure.

Members

The members are defined as follows:

LPMAC_TOKEN	A pointer to the MAC_TOKEN structure containing the token ID and parameter information.
LPLPMAC_RETURNVAL	A pointer to an LPMAC_RETURNVAL for storing the token return value information.

A Revision History

The following table lists changes that were made to the GroupWise Tokens documentation (in reverse chronological order):

Release Date	Changes
November 2012	Reviewed and updated for use with GroupWise 2012.
December 2011	Added DialPeople() (page 188).
March 2009	Added support for GroupWise 8.0 to documentation.
June 2007	Added a note that the month value of FilterSetDateAbsolute() (page 336) is from 0-11 (not 1-12).
	Updated the parameter descriptions of PrefSignature() (page 663).
October 2006	Added information about ensuring that the main window has focus before calling StatusBarLockText() (page 847) and StatusBarSetText() (page 848).
June 2006	Updated the description of the folder parameter of DisplaySettingsSetEx3() (page 221).
	Removed a duplicate link from the March 1, 2006 Revision History entry.
March 2006	Added the following tokens for GroupWise 7.0 and later: AutoSizeAllDayEventPane() (page 114), CalendarBackgroundColorDlg() (page 125), CalendarCreateDlg() (page 126), DisplaySettingsSetEx3() (page 221), FullFolderList() (page 405), ItemSetAllDayEvent() (page 523), ListViewPanels() (page 551), ListViewShowGroupHeaders() (page 552), ListViewSummary() (page 553), MessengerLaunch() (page 558), MessengerSignOff() (page 562), ManageCalendarsDlg() (page 555), ModeOffline() (page 564), PrefAppearance() (page 644), QuickViewerBottom() (page 704), QuickViewerRight() (page 706), ShowMessageSourceTab() (page 837), ShowNavBar() (page 838), SimpleFolderList() (page 840), and ViewHideCaption() (page 891).
	Updated ViewHideMenu() (page 892).
October 2005	Transitioned to revised Novell documentation standards.
	Added GetAppearance() (page 409).
March 2005	Removed PrintCalendar.
October 2004	Added the folder path syntax to RuleAddActionMoveToFolder() (page 752).
	Removed the Speller token (because it hasn't been supported since GroupWise 5.2).
June 2004	Added information to Execute() (page 58) on how to use the GroupWise Token component with .NET.
	Added an example to ItemSetText() (page 529).
	Added note about turning off the signature option of PrefSignature() (page 663) to the Remarks section.

Release Date	Changes
February 2004	Updated the syntax of JunkMailSettings() (page 540). Removed Author, Creator, and NamedField possibilities from the Field parameter of ItemSetText() (page 529). Updated links to Unsupported NDK components.
October 2003	Updated the description of ItemSaveInfo() (page 515) to state that it does not work for draft messages. Updated the Remarks section of AddressListCreateFromGroup() (page 82) about the Name Completion Search Order dialog. Updated "TPH Interface Requirements" on page 23 to show that DLLs can now be registered under both HKEY_LOCAL_MACHINE and HKEY_CURRENT_USER. Removed Author, Creator, and NamedField from ItemGetText() (page 487). These enumerated values never worked. Removed AddressBookSearch, AddressBookSearchBegin, AddressBookSearchNext, AddressBookSearchNext, AddressBookSearchPrevious, AddressBookSearchEnd, and OleAttachInsertObject. These functions were never implemented. Made several editing changes.
June 2003	Removed section on GroupWise Release Dates (because it contained information from as far back as 1997). Updated table formats for readability.
March 2003	Added documentation for the following GroupWise 6.5 tokens: AddNewChecklistItem() (page 66), AttachmentViewSame() (page 112), ButtonBarReset() (page 120), CalendarIsShowingAppts() (page 132), CalendarIsShowingNotes() (page 133), CalendarIsShowingTasks() (page 134), CalendarShowAppts() (page 135), CalendarShowNotes() (page 137), CalendarShowTasks() (page 138), CategoriesSetDlg() (page 145), CollapseFolder() (page 151), DisplaySettingsSetEx2() (page 211), ExpandFolders() (page 309), FilterCategory() (page 318), HelpUsersGuide() (page 416), HelpWhatsNew() (page 417), ItemChecklistMove() (page 448), ItemChecklistMoveDown() (page 449), ItemChecklistMoveToBottom() (page 452), ItemChecklistMoveToTop() (page 453), ItemChecklistMoveUp() (page 454), ItemNextTabSelect() (page 503), ItemPost() (page 505), ItemShowInChecklist() (page 531), JMAddUserToPAB() (page 535), JMBlockSender() (page 536), JMJunkSender() (page 537), JMTrustSender() (page 538), JunkMailHandling() (page 539), JunkMailSettings() (page 540), ListViewChecklist() (page 546), MessengerSendIM() (page 559), MessengerShowContacts() (page 560), MessengerShowPreferences() (page 561), NewAppointmentToContacts() (page 573), NewContact() (page 574), NewGroup() (page 575), NewMailToContacts() (page 577), NewNoteToContacts() (page 579), NewOrganization() (page 580), NewPhoneToContacts() (page 582), NewResource() (page 583), NewTaskToContacts() (page 586), QvButtonBarHide() (page 707), QvButtonBarShow() (page 708), RemoteRequestContacts() (page 722), ShowLongFolderList() (page 836), ViewContacts() (page 880), ViewContactsOnly() (page 881), ViewGroups() (page 889), ViewGroupsOnly() (page 890), ViewHideNonChecklistItems() (page 893), ViewOrganizations() (page 904), ViewOrganizationsOnly() (page 905), ViewResources() (page 911), and ViewResourcesOnly() (page 912).

Release Date	Changes
September 2002	Changed the possible values for the IsGroupItem parameter of ViewOpen() (page 899).
	Updated the parameter descriptions for FilterSetText() (page 345).
May 2002	Updated almost all documentation: updated syntaxes, added enumeration values, fixed type inconsistencies, added descriptions, fixed token names, etc.
	Added documentation for CachingRefresh() (page 123), CachingRetrieveProperties() (page 124), DeleteAndEmpty() (page 183), EmailImport() (page 258), FrameContentIsVisible() (page 403), FrameIsVisible() (page 404), GetSetting() (page 412), HelpCoolSolutions() (page 414), InvokeSpellerForWindow() (page 425), ItemAcceptOpenItem() (page 427), ItemChangeToAppointment() (page 443), ItemChangeToMail() (page 444), ItemChangeToNote() (page 445), ItemChangeToPhone() (page 446), ItemChangeToTask() (page 447), ItemCompleteOpenItem() (page 457), ItemForwardFlat() (page 474), ItemInfo() (page 490), ItemReplyOpenItem() (page 512), ModeCaching() (page 563), ModeOnline() (page 565), ModeRemote() (page 566), NewsGroupsNNTPDlg() (page 584), NNTPCancel() (page 590), NNTPCollapseAll() (page 591), NNTPExpandAll() (page 592), NNTPMarkAllRead() (page 593), NNTPSearch() (page 594), NNTPThreadChild() (page 595), NNTPThreadCollapse() (page 596), NNTPThreadDelete() (page 597), NNTPThreadExpand() (page 598), NNTPThreadIgnore() (page 599), NNTPThreadMarkRead() (page 600), NNTPThreadParent() (page 601), NNTPThreadWatch() (page 602), Properties() (page 679), RemoteGWCheck() (page 717), ResetNNTPFolder() (page 736), SelectTimezoneDlg() (page 784), SendNNTP() (page 796), SortFolders() (page 843), StatusBarGetText() (page 846), StatusBarLockText() (page 847), StatusBarSetText() (page 848), and StatusWindowShow() (page 849).
	Removed documentation for HelpHowDoI.
	Updated the changed functionality of ItemAccept() (page 426), ItemComplete() (page 456), and ItemReply() (page 511).
February 2002	Updated almost all documentation.
	Added solutions to premature token processing to “Why Use Tokens” on page 20.
	Fixed links to the unsupported (http://developer.novell.com/ndk/unsupported.htm) area on the NDK.
September 2001	Added text alternatives for figures.
	Removed AttachmentAddEncapsulatedItem from documentation and indicated support for GroupWise 6.x.
	Fixed typographical errors and added bullet lists for nested topics.
June 2001	Added additional explanation and example to FilterGroupMarker() (page 326).
	Made minor format improvements.
	Updated documentation to improve accessibility.
February 2001	Fixed typographical errors in FilterSetDateAbsolute() (page 336).
	Updated “Tokens and Message ID” on page 21 to state that the messageID structure is used only by the MAPI provider code.
	Reformatted all reference documentation.

Release Date	Changes
September 2000	Documented the 32KB limit for message body text. Also reformatted parameters for consistency,
July 2000	Resolved many differences between the documentation and current GroupWise 5.5 EP SP1 source files in the "Tokens Reference (A-K)" on page 61. Changes included adding missing descriptions, making corrections to syntax example, and documenting the following tokens: AccountsDlg() (page 65) AccountSync() (page 67) AddressListCreateFromGroup() (page 82) AddressListIsInSection() (page 93) AutoPilotToggle() (page 113) DocumentMassOperationDlg() (page 242) EnvClearChangedFlag() (page 265) EnvDefaultConnectionName() (page 269) EnvDocumentGetIntegrations() (page 270) EnvEditorStyle() (page 271) EnvIsCommandValid() (page 274) EnvTextCurrentCharIndex() (page 290) EnvTextCurrentLineIndex() (page 291) EnvTextCurrentWord() (page 292) EnvTextInsertMode() (page 293) EnvTextLeftChar() (page 294) EnvTextRightChar() (page 297) FollowInternetLink() (page 394) ItemDelegateOpenItem() (page 464) OfficeCloseViews() (page 606) RuleExists() (page 764) ShowAttachmentWindow() (page 832) SwitchToHTMLView() (page 850) SwitchToRTFView() (page 851) TabPrevious() (page 854) TypeChar() (page 875) Revised topic "Overview" on page 19 to update for GroupWise 5.5 EP SP1. Added new topic "Tasks" on page 33 to add "How To" information to the documentation. Added Token Identifier (ID) and Token Numeric Indexes.
July 1998	Documentation added to the Novell Developer Kit.