DMAPI (Dictation Macro Application Programming Interface) Reference

IBM ViaVoice™ SDK for Windows®

Version 1.6

Printed in the USA

Note:
Before using this information and the product it supports, be sure to read the general information under Appendix A “Notices”.

First Edition (December 1999)

Contents

This book provides detailed information on developing Windows® 95/98 or Windows NT™ speech-aware applications using IBM ViaVoice™ Speech Solutions Developer's Kit (SDK) for Windows and Speech Manager Application Programming Interfaces (SMAPI) from IBM. The ViaVoice SDK provides three sets of APIs: Speech Manager APIs (SMAPI), Dictation Macro APIs (DMAPI), and Grammar Compiler APIs.

This books covers the DMAPI functions and is prepared in Portable Document Format (PDF) to provide the advantages of text search and cross-reference hyperlinking. It is viewable with the Adobe Acrobat Reader v.3.x. We recommend that you print all or part of this guide for quick reference.

Who Should Read This Book

Read this book if you are a software developer interested in writing Windows 95/98 or Windows NT 4.0 applications using the ViaVoice SDK APIs.

Related Publications

Refer to the following publications included with this version for additional programming, reference, and design information:

- SMAPI Reference
- SMAPI Developer's Guide
- SAPI Reference
- ActiveX Developer's Guide

Refer to the following sources for additional programming, reference, and design information:

- ViaVoice Developer's Corner website at:
  http://www.ibm.com/viavoice/dev_home.html

- IBM ViaVoice SDK Web Channel at:
  http://www.software.ibm.com/viavoice/subscribe.html
- OLE Automation Reference from Microsoft

Chapter 1 Application Programming Interface

The Dictation Macro API (DMAPI) is a set of C-language functions that allows developers to access dictation macros and dictation templates within their applications. The DMAPI is intended for use in SMAPI dictation applications written for ViaVoice. A good understanding of the ViaVoice Speech Manager API (SMAPI) is assumed.

A dictation macro is a shortcut for entering text. A dictation macro consists of a name and an action. The name is the voice command that is spoken to invoke the action. When the macro name is spoken, the dictation macro is replaced by the text defined for the macro. For example, a macro called "outside-address" could be defined with an action of "IBM Corporation, White Plains, NY, USA." Whenever "outside-address" is spoken, it is replaced with "IBM Corporation, White Plans, NY, USA."

A dictation template is similar to a dictation macro, but a template might contain predefined fields into which the user can dictate. In a sense, a template is a form that can be filled in by speech.

Users create dictation macros and templates through the Dictation Macro Editor. The Dictation Macro Editor is included in the ViaVoice Run Time Kit.

An application handles dictation macros and templates differently. Macros are added to the dictation text vocabulary, while templates are included as a dynamic command vocabulary (since templates must be expanded immediately and only dynamic command vocabularies are immediately firmed up by the speech engine). Also, there is additional field-related processing that must be performed for templates.

DMAPI is provided as a separate DLL from SMAPI. To compile applications that use DMAPI, simply include the header file, DMAPI.H, which contains all of the necessary definitions. Be sure to include DMAPI.H after the standard Windows include files. Also, depending on your build environment, you might have to set the INCLUDE environment variable to point to the path where DMAPI.H is located or otherwise specify this path (such as on the compiler command line).

DMAPI is supported in all languages in which ViaVoice is provided. All user-visible elements are translated (for example, the keywords for macro expansions). Various DMAPI files and DLLs have different paths and names according to the language. Currently, the following languages are available:

Table 1. Languages Currently Available

Language	Language ID	DLL	Macro File
U.S. English	En_US	dmapi_us.dll	<installdir>\vocabs\langs\En_US\macros\allus.dct
U.K. English	En_UK	dmapi_uk.dll	<installdir>vocabs\langs\En_UK\macros\alluk.dct
German	Gr_GR	dmapi_gr.dll	<installdir>\vocabs\langs\Gr_GR\macros\allgr.dct
French	Fr_FR	dmapi_fr.dll	<installdir>\vocabs\langs\Fr_FR\macros\allfr.dct
Italian	It_IT	dmapi_it.dll	<installdir>\vocabs\langs\It_IT\macros\allit.dct
Spanish	Es_ES	dmapi_es.dll	<installdir>\vocabs\langs\Es_ES\macros\alles.dct
Arabic	Ar_AR	dmapi_ar.dll	<installdir>\vocabs\langs\Ar_AR\macros\allar.dct
Japanese	Ja_JP	dmapi_jp.dll	<installdir>\vocabs\langs\Ja_JP\macros\alljp.dct
Chinese (Simplified)	Zh_CN	dmapi_cn.dll	<installdir>\vocabs\langs\Zh_CN\macros\allcn.dct
Chinese (Traditional)	Zh_TW	dmapi_tw.dll	<installdir>\vocabs\langs\Zh_TW\macros\alltw.dct

The ViaVoice Run Time Kit provides a set of predefined dictation macros for each language and the DLLs necessary to call the DMAPI functions. The following files are used at run time by DMAPI:

- dmapi.dll - This is the DLL that contains all the language-independent functions. It is located in <installdir>\bin.
- dmapi_xx.dll - This is a DLL that contains language-specific functions (such as keywords for macro expansions). This is located in <installdir>\bin.
- *.dct - These files contain the dictation macro definitions. These files are located in <installdir>\vocabs\langs\xx_XX\macros and <installdir>\users\<username>.

If you need additional macros for your application at run time, you must prepare and export these macros using the Dictation Macro Editor. This will produce a .DCT file. You must also instruct users to import this file with the Dictation Macro Editor before they can use your application.

Notes:

Currently, DMAPI only allows applications to open a read session with the DMAPI. This means that applications can query and use dictation macros and templates, but cannot create new macros or templates. To create new macros and templates, the Dictation Macro Editor must be used.

DMAPI is not thread-save; that is, you should always call DMAPI functions from the same thread of your application. Otherwise, you will have to handle the synchronization yourself.DMAPI offers functions to perform the following tasks:

- Initializing DMAPI
- Getting macro and template definitions
- Querying the current set of macros and templates
- Extracting information from macro actions and expansion text
- Updating the application's internal macro database
- Handling errors
- Closing DMAPI

Typically, speech-aware applications provide a text window (such as a rich-text control) into which the user dictates text. The user can dictate as well as type into this area. These applications should perform the following functions:

- When starting up:
- Initialize DMAPI.
- Query the current set of macros and add them to the text vocabulary.
- Query the current set of templates and create a dynamic command vocabulary for these.
- Call **DmUpdate** before switching the recognition engine to dictation to ensure that the most recent definitions of dictation macros and templates are active.
- Call **DmGetMacro** and **DmGetTemplate** whenever a firm word is received from the speech engine and expand it if it is a macro or template.
- When terminating, close DMAPI.

Initializing DMAPI

Use **DmOpen** to initialize DMAPI. As input parameters, you must specify the user ID, domain, and language (you should use the appropriate SMAPI calls to obtain the default values). **DmOpen** returns a handle to the macro database. This handle is used when making all other DMAPI calls.

DmOpen performs the following functions:

- Loads the appropriate DMAPI_xx.DLL for the language specified.
- Loads the domain-specific dictation macros from the file <domain name>xx.dct. (First <installdir>\users\<username> is searched and then <installdir>\vocabs\langs\xx_XX\macros is searched.)
- Loads the dictation macros from the file allxx.dct for the language specified. (First <installdir>\users\<username> is searched and then <installdir>\vocabs\langs\xx_XX\macros is searched.)

Getting Macro Definitions

Whenever a firm word is received from the speech engine, use **DmGetMacro** and check the return code to determine if the recognized word is a dictation macro. Note that for some types of words, you get an indication that they are macros, although they are not contained in the dictation macro files (*.DCT.) This is done to reduce special handling in typical applications. The following “normal” words are returned as “pseudo-macros”:

- All numbers (the digit flag is set)
- For Italian only, all words ending with apostrophe (the join right flag is set)

DMAPI returns a macro structure (DM_MACRO_STRUCT), which contains the name of the macro and the macro text. It contains additional information as well, such as formatting flags and domain name.

With **DmGetMacro**, you can specify a flag which indicates the type of expansion to perform for the macro. Most applications will want to use the normal expansion flag and immediately show the expanded text in the application window. Note that if the macro contains a program link (either a DLL

or a system call), this call is executed within the **DmGetMacro** function. There is a two-second time-out for the call. If the call runs for more than two seconds, you will receive an error code from **DmGetMacro**.

DMAPI allocates memory for the macro structures, so your application should use **DmFreeMacro** to free this memory.

Getting Template Definitions

Templates are handled similarly to dictation macros. There are some additional considerations:

- Templates should expand immediately, so these words should not be in the text vocabulary. Applications should create a dynamic command vocabulary for templates.
- Templates usually contain fields into which the user dictates. The **DmGetTemplate** call returns a pointer to a chain of fields with additional information about each template field (such as position of the field within the template, the type of field, and its default text.)

Applications that support templates should perform the following functions:

- Whenever a firm word is received from the speech engine, use **DmGetTemplate** and check the return code to determine if the recognized word is a template. Expand the template immediately, and display the text to the user.
- Position the cursor in the first field.
- Enable a dynamic vocabulary with the commands “next field” and “previous field,” so that the user can navigate between fields using speech. Depending on the type of field, do the following:
    - For text fields, leave the dictation text vocabulary active (or enable it, if necessary) and let the user dictate into the field.
    - For spell fields, temporarily disable the dictation text vocabulary and enable a dynamic command vocabulary for a spelling alphabet.
    - For digit fields, temporarily disable the dictation text vocabulary and enable a dynamic command vocabulary for digits.
    - For selection fields, temporarily disable the dictation text vocabulary and enable a dynamic command vocabulary for the list of commands contained in the template field structure (DM_TEMPLATE_FIELD.)

- When the user is finished working with the template, enable the dictation text vocabulary.

DMAPI allocates memory for the macro structures, so your application should use **DmFreeMacro** to free this memory.

Querying the Current Set of Macros and Templates

Use **DmQueryMacroNames** and **DmQueryTemplateNames** to get a list of all current dictation macros and templates. Your application needs this to:

- Add macro names to the text vocabulary
- Define a dynamic command vocabulary for templates
- Present a list of available macros and templates to the user (without the “pseudo-macros,” of course)

DMAPI allocates memory for these lists, so your application should use **DmFreeNames** to free this memory.

Extracting Information From Macro Actions and Expansion Text

Some applications might want to parse the actions of dictation macros and templates before actually expanding them. With DMAPI, you can extract information from macro actions and expansion text. To do so, first use **DmGetMacro** with the expand flag set to DM_EXPAND_NONE. Parse the action with the **DmQueryDelimiters** and **DmTypeFromKeyword** calls. **DmGetMacro** can then be called a second time with the expand flag set to DM_EXPAND_NORMAL, if necessary.

Updating the Application's Internal Macro Database

DMAPI maintains a dictation macro and template database across all speech-aware applications running on the system. For example, the Dictation Macro Editor could be running (where the user creates macros and templates.) At the same time, multiple dictation applications (where the user dictates text using these macros and templates) could also be running. DMAPI maintains a copy of the database for each application. Whenever an application calls **DmUpdate**, its copy will be refreshed. Typically, dictation applications will want to call **DmUpdate** before switching the engine to dictation.

Handling Errors

Use **DmGetLastError** whenever you receive a return code other than DM_ERR_OK from DMAPI. This provides your application with additional information about the error that occurred. It also provides a message that your application might want to present to the user.

Closing DMAPI

Use **DmClose** to release all resources held by DMAPI. After calling DmClose, you can no longer use DMAPI unless you call DmOpen again.

Chapter 2

DMAPI Functions

This chapter describes the format of the Dictation Macro (DMAPI) function calls and contains descriptions for each of the DMAPI functions.

Format of the Function Call Descriptions

The description of each function call contains the following information:

Function Name

The name of the function call.

Purpose

The purpose and description of the function call.

Syntax

The syntax of the function as declared in DMAPI.H.

Parameters

Definitions of the parameters.

Return Values

Return values that are set by DMAPI.

For detailed descriptions of DMAPI errors, see “DMAPI Return Codes and Messages” on page 65.

DMAPI Function Calls

The following section lists all of the DMAPI function calls.

- DmClose
- DmCreateMacro
- DmDestroyMacro
- DmFlushEx
- DmFreeMacro
- DmFreeMacroEx
- DmFreeNames
- DmGetGroupsOnPage
- DmGetKey
- DmGetKeyGroup
- DmGetKeyGroupOption
- DmGetKeyGroupOptionSelection
- DmGetLastError
- DmGetMacro
- DmGetMacroEx
- DmGetNumberOfGroups
- DmGetNumberOfOptions
- DmGetNumberOfSelections
- DmGetTemplate
- DmIsNumberFlagged
- DmKeywordFromType
- DmOpen
- DmQueryDelimiters
- DmQueryMacroNames
- DmQueryTemplateNames
- DmQuerySetAndStoreKey
- DmSetKey
- DmStartMacroEditor

- DmStoreKey
- DmTypeFromKeyword
- DmUpdate

DmClose

Closes dictation macros for the specified user, domain, and language.

Syntax

```
unsigned long DmClose (DMHANDLE hmac);
```

Parameters

hmac
input - The DMAPI handle for the set of macros to be closed.

Return Values

DM_ERR_INVALID_HANDLE
An invalid macro handle was passed to DMAPI.

DM_ERR_OK
No error occurred.

Remarks

It also frees internally-allocated resources.

DmCreateMacro

Creates new macro definition for the current set of macros.

Syntax

```
unsigned long DmCreateMacro (DMHANDLE hmac, PDM_MACRO_STRUCT macro);
```

Parameters

hmac
Input. Macro handle.

macro
Input. Macro definition to be stored in the current set of macros.

Return Values

DM_ERR_OK
No error occurred.

DM_ERR_INVALID_HANDLE
An invalid macro handle was passed to DMAPI.

DM_ERR_MALLOC
A memory allocation failure occurred.

DM_ERR_SYSTEM_PARM_LONG
A parameter specified in a system call is too long.

DmDestroyMacro

Removes a macro definition from the current set of macros.

Syntax

```
unsigned long DmDestroyMacro (DMHANDLE hmac, char * name, char * domain,
unsigned long type);
```

Parameters

hmac
Input. Macro handle.

name
Input. The domain name in which the macro resides.

domain
Input. The domain name in which the macro resides.

type
Input. The macro type (DM_MACTYPE_MACRO, DM_MACTYPE_TEMPLATE)

Return Values

DM_ERR_OK
No error occurred.

DM_ERR_INVALID_HANDLE
An invalid macro handle was passed to DMAPI.

DM_ERR_MALLOC
A memory allocation failure occurred.

DM_ERR_SYSTEM_PARM_LONG
A parameter specified in a system call is too long.

DmFlushEx

Makes the number filter firm up on stacked data.

Syntax

```
unsigned long DmFlushEx (DMHANDLE hmac, unsigned short uExpand,
PDM_MACRO_STRUCT *pmac, unsigned long *num);
```

Parameters

hmac
input - A macro handle.

uExpand
input - One of the following flags:

DM_EXPAND_NONE - No expansion, just returns the text as it is stored.

DM_EXPAND_NORMAL - Expands the macro and all embedded macros with imm_expand set If the macro itself is a delayed macro, the text will contain the macro name in the form

<MACRO macroname>.

DM_EXPAND_FULL - Expands the macro and all embedded macros no matter how the imm_expand flags are set.

pmac
output - A pointer to the macro definition of the word was found or NULL in case of error. If NULL is returned, the return code gives information about the type of error. Use **DmFreeMacro** to free memory used by pointers returned by this function.

num
output - Number of pmac structures returned.

Return Values

DM_ERR_OK
No error occurred.

DM_ERR_MALLOC
A memory allocation failure occurred.

DmFreeMacro

Frees memory associated with a macro or template previously obtained by **DmGetMacro** or **DmGetTemplate**.

Syntax

```
void DmFreeMacro (PDM_MACRO_STRUCT  macro);
```

Parameters

macro
input - A pointer to the macro definition.

Return Values

None.

DmFreeMacroEx

Frees memory of a macro or template previously obtained by **DmGetMacro** or **DmGetTemplate**.

Syntax

```
void DmFreeMacroEx (PDM_MACRO_STRUCT macro, unsigned long num);
```

Parameters

macro
input - A pointer to the macro definition.

num
input - Number of macros in the macro. (See DmGetMacroEx)

Return Values

None.

DmFreeNames

Frees memory associated with the fields previously allocated by **DmQueryMacroNames** or **DmQueryTemplateNames**.

Syntax

```
void DmFreeNames (unsigned short number, char **field);
```

Parameters

number
input - The number of list entries.

field
input - A pointer to a field of strings that contains the names of macros or templates.

Return Values

None.

DmGetGroupsOnPage

Returns the number of groups to display per options page.

Syntax

```
unsigned long DmGetGroupsOnPage (DMHANDLE hmac, int *page1, int *page2,
int *page3);
```

Parameters

hmac
input - A macro handle.

page1 to *page3*
output - The number of groups to display on the Options page n, if all are 0, do not show an options panel at all.

Return Values

DM_ERR_OK
No error occurred.

DmGetKey

Queries the number filter for this value to a key.

Syntax

```
unsigned long DmGetKey (DMHANDLE  hmac, char *key, char **value,
unsigned long *num);
```

Parameters

hmac
input - The macro handle.

key
input - The key.

value
output - The value.

num
output - The number of values returned.

Return Values

DM_ERR_OK
No error occurred.

DM_ERR_MALLOC
A memory allocation failure occurred.

DM_ERR_INVALID_KEYWORD
An invalid keyword was passed to DMAPI.

DmGetKeyGroup

Returns the string to display for a given group number.

Syntax

```
unsigned long DmGetKeyGroup (DMHANDLE hmac, int group,
char **display_string);
```

Parameters

hmac
input - The macro handle.

group
input - The group number for which the display_string should be returned.

display_string
output - The ANSI string to display on the Options panel.

Return Values

DM_ERR_OK
No error occurred.

DM_ERR_INVALID_KEYWORD
An invalid keyword was passed to DMAPI.

DmGetKeyGroupOption

Returns the string to display for a given group number/option.

Syntax

```
unsigned long DmGetKeyGroupOption (DMHANDLE hmac, int group, int option,
char **display_string);
```

Parameters

hmac
input - The macro handle.

group
input - The group number for which the display_string should be returned.

option
input - The option number for which the display_string should be returned.

display_string
output - The ANSI string to display on the Options panel.

Return Values

DM_ERR_OK
No error occurred.

DM_ERR_INVALID_KEYWORD
An invalid keyword was passed to DMAPI.

DmGetKeyGroupOptionSelection

Returns the string to display for a given group number/option/selection.

Syntax

```
unsigned long DmGetKeyGroupOptionSelection (DMHANDLE hmac, int group,
int option, int selection, char **display_string);
```

Parameters

hmac
input - The macro handle.

group
input - The group number for which the display_string should be returned.

option
input - The option number for which the display_string should be returned.

selection
input - The selection number for which the display_string should be returned.

display_string
output - The ANSI string to display on the Options panel.

Return Values

DM_ERR_OK
No error occurred.

DM_ERR_INVALID_KEYWORD
An invalid keyword was passed to DMAPI.

DmGetLastError

Returns additional information about the last error.

Syntax

```
void DmGetLastError (PDMERROR  errbuf);
```

Parameters

errbuf
output - A pointer to a DMERROR structure.

Return Values

None.

Remarks

This call does not change the internally-stored error information. **DmGetLastError** uses a **DMERROR** structure which contains the following information:
errbuf->errno
The DM_ERR_... value of the last error.
errbuf->string
Additional information about the last error:

```
DM_ERR_ACT_FILE_PARSE          line number
DM_ERR_BAD_EXE_FORMAT          file name
DM_ERR_DUPLICATE_MACRO         macro name
DM_ERR_EACCESS                 file name
DM_ERR_EBADF                   file name
DM_ERR_EMFILE                  file name
DM_ERR_ENOENT                  file name
DM_ERR_EXPDLL_LOAD_FAILED      DLL name
DM_ERR_EXPDLL_QUERYFUN_FAILED  function name
DM_ERR_FILE_READ               file name
DM_ERR_INVALID_DOMAIN          domain name
DM_ERR_INVALID_LANGUAGE        language code
```

```
DM_ERR_INVALID_MACRORELEASE   release number
DM_ERR_INVALID_MACROVERSION   version number
DM_ERR_INVALID_USERID         user ID
DM_ERR_LOAD_RESOURCE_DLL      name of DLL
DM_ERR_MACRO_NOT_FOUND        macro name
DM_ERR_TEMPLATE_NOT_FOUND     template name
DM_ERR_UNDEFINED              file name
```

DmGetMacro

Gets a macro definition from the current set of macros.

Syntax

```
unsigned long DmGetMacro(DMHANDLE hmac, char *word, unsigned short uExpand,
PDM_MACRO_STRUCT *pmac);
```

Parameters

hmac
input - The macro handle.

word
input - The name of the macro.

uExpand
input - The type of expansion to be performed, which can be one of the following:

DM_EXPAND_NONE - The macro should not be expanded, only the text as it is stored will be returned.

DM_EXPAND_NORMAL - The macro and all embedded macros with **imm_expand** set to TRUE will be expanded. Delayed macros will not be expanded. If the macro itself is a delayed macro, the text will contain the macro name in the form <MACRO macroname>.

DM_EXPAND_FULL - The macro and all embedded macros (regardless of the value of imm_expand) will be expanded.

pmac
output - If the word was found, a pointer to the macro definition; otherwise, it will be set to NULL. If NULL is returned, the return code gives information about the type of error. Use **DmFreeMacro** to free memory used by pointers returned by this function.

Return Values

DM_ERR_EXPDLL_ERR
An error occurred executing a DLL function.

DM_ERR_EXPDLL_LOAD_FAILED
: DMAPI was unable to load a DLL.

DM_ERR_EXPDLL_QUERYFUN_FAILED
: DMAPI was unable to load a function of a DLL.

DM_ERR_EXPDLL_TIMEOUT
: The execution of a DLL function timed out.

DM_ERR_INVALID_HANDLE
: An invalid macro handle was passed to DMAPI.

DM_ERR_MACRO_NOT_FOUND
: DMAPI was unable to find the specified macro. Either a macro does not exist by that name, or you specified an incorrect name.

DM_ERR_MALLOC
: A memory allocation failure occurred.

DM_ERR_OK
: No error occurred.

DM_ERR_SYSTEM_PARM_LONG
: A parameter specified in a system call is too long.

DmGetMacroEx

Gets a macro definition from the current set of macros.

Syntax

```
unsigned long DmGetMacroEx (DMHANDLE hmac, char *word, char *soundslike,
unsigned long flags, unsigned long tag, unsigned short uExpand,
PDM_MACRO_STRUCT *pmac);
```

Parameters

hmac
input - The macro handle.

word
input - The name of the macro.

soundslike
input - The soundslike field for spellout function.

flags
input - The number and other flags.

uExpand
input - The type of expansion to be performed, which can be one of the following:

DM_EXPAND_NONE - The macro should not be expanded, only the text as it is stored will be returned.

DM_EXPAND_NORMAL - The macro and all embedded macros with **imm_expand** set to TRUE will be expanded. Delayed macros will not be expanded. If the macro itself is a delayed macro, the text will contain the macro name in the form <MACRO macroname>.

DM_EXPAND_FULL - The macro and all embedded macros (regardless of the value of imm_expand) will be expanded.

pmac
output - If the word was found, a pointer to the macro definition; otherwise, it will be set to NULL. If NULL is returned, the return code gives information about the type of error. Use **DmFreeMacro** to free memory used by pointers returned by this function.

Return Values

DM_ERR_EXPDLL_ERR
: An error occurred executing a DLL function.

DM_ERR_EXPDLL_LOAD_FAILED
: DMAPI was unable to load a DLL.

DM_ERR_EXPDLL_QUERYFUN_FAILED
: DMAPI was unable to load a function of a DLL.

DM_ERR_EXPDLL_TIMEOUT
: The execution of a DLL function timed out.

DM_ERR_INVALID_HANDLE
: An invalid macro handle was passed to DMAPI.

DM_ERR_MACRO_NOT_FOUND
: DMAPI was unable to find the specified macro. Either a macro does not exist by that name, or you specified an incorrect name.

DM_ERR_MALLOC
: A memory allocation failure occurred.

DM_ERR_OK
: No error occurred.

DM_ERR_SYSTEM_PARM_LONG
: A parameter specified in a system call is too long.

DmGetNumberOfGroups

Returns the number of groups for display in the Options panel.

Syntax

```
unsigned long DmGetNumberOfGroups (DMHANDLE hmac);
```

Parameters

hmac
input - The macro handle.

Return Values

0
When no groups are found.

>0
Represents the number of groups found.

DmGetNumberOfOptions

Returns the number of options for display in the Options panel.

Syntax

```
unsigned long DmGetNumberOfOptions (DMHANDLE hmac, int group);
```

Parameters

hmac
input - The macro handle.

group
input - The group number for which the options should be returned.

Return Values

0
When no options are found.

>0
Represents the number of options found.

DmGetNumberOfSelections

Returns the number of selections for display in the Options panel.

Syntax

```
unsigned long DmGetNumberOfSelections (DMHANDLE hmac, int group,
int option);
```

Parameters

hmac
input - The macro handle.

group
input - The group number for which the selection should be returned.

option
input - The option number for which the selection should be returned.

Return Values

0
When no selections are found.

>0
Represents the number of selections found.

DmGetTemplate

Gets a template definition from the current set of templates.

Syntax

```
unsigned long DmGetTemplate (DMHANDLE hmac, char *word,
unsigned short uExpand, PDM_MACRO_STRUCT *pmac);
```

Parameters

hmac
Input - The DMAPI handle for the current set of templates.

word
input - The name of the template.

uExpand
input - The type of expansion to be performed, which can be one of the following:

DM_EXPAND_NONE - The template should not be expanded, only the text as it stored will be returned.

DM_EXPAND_NORMAL - The template and all embedded macros with **imm_expand** set to TRUE will be expanded. Delayed macros will not be expanded.

DM_EXPAND_FULL - The template and all embedded macros (regardless of the value of imm_expand) will be expanded.

pmac
output - If the word was found, a pointer to the template definition; otherwise, it will be set to NULL. If NULL is returned, the return code gives information about the type of error. Use **DmFreeMacro** to free memory used by pointers returned by this function.

Return Values

DM_ERR_EXPDLL_ERR
An error occurred executing a DLL function.

DM_ERR_EXPDLL_LOAD_FAILED
DMAPI was unable to load a DLL.

DM_ERR_EXPDLL_QUERYFUN_FAILED
: DMAPI was unable to load a function of a DLL.

DM_ERR_EXPDLL_TIMEOUT
: The execution of a DLL function timed out.

DM_ERR_INVALID_HANDLE
: An invalid macro handle was passed to DMAPI.

DM_ERR_MALLOC
: A memory allocation failure occurred.

DM_ERR_OK
: No error occurred.

DM_ERR_SYSTEM_PARM_LONG
: A parameter specified in a system call is too long.

DM_ERR_TEMPLATE_NOT_FOUND
: DMAPI was unable to find the specified template. Either a template does not exist by that name, or you specified an incorrect name.

DmIsNumberFlagged

Returns !=0 if the flag is a number flag.

Syntax

```
unsigned long DmIsNumberFlagged (unsigned long flags);
```

Parameters

flags
input - The flag returned from the speech engine.

Return Values

0
Not flagged as a number

>0
Bits 0-11 contain the number flags

DmKeywordFromType

Gets the keywords used by the Dictation Macro Editor.

Syntax

```
unsigned long DmKeyWordFromType (DMHANDLE dmac, unsigned short type,
char **keyword);
```

Parameters

dmac

input - The macro handle.

type

input - The type from which to get the keyword, which can be one of the KEYBOARD_xx found below:

KEYWORD_INVALID	0
KEYWORD_NONE	1
KEYWORD_DAYOFWEEK	2
KEYWORD_DATE	3
KEYWORD_TIME	4
KEYWORD_DAY	5
KEYWORD_DAYORDINAL	6
KEYWORD_MONTHOFYEAR	7
KEYWORD_MONTHSHORT	8
KEYWORD_MONTH	9
KEYWORD_YEAR	10
KEYWORD_YEARSHORT	11
KEYWORD_MSGSTRING	14
KEYWORD_DLLCALL	15
KEYWORD_DLLSTRING	16
KEYWORD_SYSTEM	17

```
KEYWORD_MACRO          18
```

keyword
The keyword of the input type. The language of the keyword depends on the language associated with the macro handle.

Return Values

DM_ERR_INVALID_KEYTYPE
An invalid key type was passed to DMAPI.

DM_ERR_OK
No error occurred.

DmOpen

Opens dictation macros for the specified user, domain, and language.

Syntax

```
unsigned long DmOpen (void *init, char *username, char *domain, char *lang,
PDMHANDLE hmac);
```

Parameters

init
input - This parameter is RESERVED and must be set to NULL.

username
input - The user name (5-char. code) for which the macros should be initialized.

domain
input - The domain (5-char. code) for which the macros should be initialized.

lang
input - The language (5-char. code) for which the macros should be initialized. This code should be retrieved from the speech recognition engine.

hmac
output - The Dictation Macro Editor specific handle which is needed for further calls for this set of macros.

Return Values

DM_ERR_DUPLICATE_MACRO
An attempt was made to load an already existing macro.

DM_ERR_EACCESS
An error occurred attempting to access a macro file. Either the specified file is a directory, or you attempted to write to a read-only file.

DM_ERR_EBADF
The file is not opened.

DM_ERR_EMFILE
No more file handles are available.

DM_ERR_ENOENT
The specified file or path was not found.

DM_ERR_INVALID_DOMAIN
An invalid domain name was passed to DMAPI.

DM_ERR_INVALID_LANGUAGE
An invalid language code was passed to DMAPI.

DM_ERR_INVALID_MACRO
An attempt was made to load an invalid macro. The macro file may be corrupted.

DM_ERR_INVALID_MACROFILE
The macro file specified is invalid.

DM_ERR_INVALID_USERID
An invalid user ID was passed to DMAPI.

DM_ERR_LOAD_RESOURCE_DLL
DMAPI was unable to load the specified DLL.

DM_ERR_MALLOC
A memory allocation failure occurred.

DM_ERR_NOTIFY_EXCEEDED
The maximum number of Dictation Macro Editor sessions has been exceeded.

DM_ERR_OK
No error occurred.

DM_ERR_UNDEFINED
An undefined file access error occurred.

DmQueryDelimiters

Requests the keyword and template field delimiters used by the Dictation Macro Editor.

Syntax

```
void DmQueryDelimiters (unsigned char *FieldStart,
unsigned char *FieldEnd, unsigned char *KeywordStart,
unsigned char *KeywordEnd, unsigned char *EscapeChar);
```

Parameters

FieldStart
output - The character which indicates the start of a field.

FieldEnd
output - The character which indicates the end of a field.

KeywordStart
output - The character which indicates the start of a keyword.

KeywordEnd
output - The character which indicates the end of a keyword.

EscapeChar
output - The character which indicates the escape character.

Return Values

None.

DmQueryMacroNames

Requests a list of the names of all available dictation macros.

Syntax

```
unsigned long DmQueryMacroNames (DMHANDLE hmac, unsigned short *number,
char ***names);
```

Parameters

hmac
input - The macro handle.

number
output - The number of entries in the list.

names
output - A pointer to a field of string pointers containing the names of the macros. If the number of macros is 0, this pointer is set to NULL.

Return Values

DM_ERR_INVALID_HANDLE
An invalid macro handle was passed to DMAPI.

DM_ERR_MALLOC
A memory allocation failure occurred.

DM_ERR_OK
No error occurred.

Remarks

Use **DmFreeNames** to free memory used by the field returned by this function.

DmQueryTemplateNames

Requests a list of the names of all available dictation templates.

Syntax

```
unsigned long DmQueryTemplateNames(DMHANDLE hmac, unsigned short *  number,
char ***names);
```

Parameters

hmac
input - The handle for the current set of templates.

number
output - The number of entries in the list.

names
output - A pointer to a field of string pointers containing the names of the templates. If the number of templates is 0, this pointer is set to NULL.

Return Values

DM_ERR_INVALID_HANDLE
An invalid macro handle was passed to DMAPI.

DM_ERR_MALLOC
A memory allocation failure occurred.

DM_ERR_OK
No error occurred.

Remarks

Use **DmFreeNames** to free memory used by the field returned by this function.

DmQuerySetAndStoreKey

Sets the current session and stores for future use the given value to the given key.

Syntax

```
unsigned long DmQuerySetAndStoreKey(DMHANDLE hmac, char *key, char *value);
```

Parameters

key
input - The number dictation key.

value
input - The value associated to the key.

Return Values

DM_ERR_INVALID_KEYWORD
An invalid keyword was passed to DMAPI.

DM_ERR_OK
No error occurred.

DmSetKey

Sets the key for the current session only.

Syntax

```
unsigned long DmSetKey (DMHANDLE hmac, char *key, char *value);
```

Parameters

hmac
input - The macro handle.

key
input - The number dictation key.

value
input - The value associated to the key.

Return Values

DM_ERR_INVALID_KEYWORD
An invalid keyword was passed to DMAPI.

DM_ERR_OK
No error occurred.

DmStartMacroEditor

Starts the Dictation Macro Editor.

Syntax

```
unsigned long DmStartMacroEditor (char *lang);
```

Parameters

lang
input - The language for which the Dictation Macro Editor should be started.

Return Values

DM_ERR_BAD_EXE_FORMAT
The specified EXE file is invalid. It may be a non-Win32 EXE, or there may be an error in the EXE image.

DM_ERR_ENOENT
The specified file or path was not found.

DM_ERR_OK
No error occurred.

DM_ERR_OUT_OF_SYSTEM_RES
The system is out of memory or other system resources.

Remarks

If the Dictation Macro Editor is already running, **DmStartMacroEditor** switches focus to the already-running Dictation Macro Editor.

DmStoreKey

Stores the given value to the given key for future use.

Syntax

```
unsigned long DmStoreKey (DMHANDLE hmac, char *key, char *value);
```

Parameters

hmac
input - The macro handle.

key
input - The number dictation key.

value
input - The value associated to the key.

Return Values

DM_ERR_INVALID_KEYWORD
An invalid keyword was passed to DMAPI.

DM_ERR_OK
No error occurred.

DmTypeFromKeyword

Returns the type constant of the specified keyword.

Syntax

```
unsigned long DmTypeFromKeyword (DMHANDLE dmac, char *keyword,
unsigned short *type);
```

Parameters

dmac
input - The handle for the current set of macros.

keyword
input - The keyword from which to get the type. The keyword depends on the language associated with the macro handle.

type
output - The type of the keyword, which can be one of the following:

KEYWORD_INVALID	0
KEYWORD_NONE	1
KEYWORD_DAYOFWEEK	2
KEYWORD_DATE	3
KEYWORD_TIME	4
KEYWORD_DAY	5
KEYWORD_DAYORDINAL	6
KEYWORD_MONTHOFYEAR	7
KEYWORD_MONTHSHORT	8
KEYWORD_MONTH	9
KEYWORD_YEAR	10
KEYWORD_YEARSHORT	11
KEYWORD_MSGSTRING	14
KEYWORD_DLLCALL	15

```
KEYWORD_DLLSTRING     16
KEYWORD_SYSTEM        17
KEYWORD_MACRO         18
```

Return Values

DM_ERR_INVALID_KEYWORD
: An invalid keyword was passed to DMAPI.

DM_ERR_OK
: No error occurred.

DmUpdate

Updates the loaded macro database with all changes that have been made to macros or templates since the last time **DmUpdate** was called.

Syntax

```
unsigned long DmUpdate (DMHANDLE hmac, int reserved1, int reserved2);
```

Parameters

hmac
input - The DME handle for the macro database to be updated.

reserved1
input - RESERVED parameter - must be set to 0.

reserved2
input - RESERVED parameter - must be set to 0.

Return Values

DM_ERR_DUPLICATE_MACRO
An attempt was made to load an already existing macro.

DM_ERR_EACCESS
An error occurred attempting to access a macro file. Either the specified file is a directory, or you attempted to write to a read-only file.

DM_ERR_EBADF
The file is not opened.

DM_ERR_EMFILE
No more file handles are available.

DM_ERR_ENOENT
The specified file or path was not found.

DM_ERR_INVALID_HANDLE
An invalid macro handle was passed to DMAPI.

DM_ERR_INVALID_MACRO
: An attempt was made to load an invalid macro. The macro file may be corrupted.

DM_ERR_INVALID_MACROFILE
: The macro file specified is invalid.

DM_ERR_NOUPDATE
: This error is returned by DmUpdate when no changes have been made to any macros or templates since the last DmUpdate call.

DM_ERR_OK
: No error occurred.

DM_ERR_UNDEFINED
: An undefined file access error occurred.

Chapter 3

Data Types

The following data types are used by DMAPI. They are listed in alphabetic order.

DMERROR

Error structure returned by DmGetLastError.

Error structure

```
typedef struct
{
ULONG        errorcode;
UCHAR        string[DM_MAX_ERROR_LEN+1];
} DMERROR, *PDMERROR;
```

Fields

errorcode
DMAPI error code.

string
Additional information based on error code.

DM_MACRO_STRUCT

DM_MACRO_STRUCT is used for both "normal" macro definitions and templates. For macros, numfield is always set to 0 and fields is always NULL. For templates, numfield is greater than 0 and fields points to the field definition(s). Additionally, imm_expand is ignored for templates, since templates are always expanded immediately.

Macro structure

```
typedef struct
{
ULONG                type;
PCHAR                name;
PCHAR                text;
PCHAR                desc;
CHAR                 domain[MAX_DOMAIN_LEN];
ULONG                joinleft;
ULONG                joinright;
ULONG                joindigit;
ULONG                casing;
ULONG                imm_expand;
ULONG                numfield;
PDM_TEMPLATE_FIELD   *fields;
ULONG                affect;
} DM_MACRO_STRUCT, *PDM_MACRO_STRUCT;
```

Fields

type
Type of macro (DM_MACTYPE_MACRO or DM_MACTYPE_TEMPLATE)

name
Word to activate macro.

text
Macro text.

desc

Template description.

domain
Domain name (length is 0 if this is a global macro).

joinleft
Formatting flag: Join to previous word.

joinright
Formatting flag: Join to next word.

joindigit
Formatting flag: Join digits.

casing
Casing of next word (DM_CASE_NONE, DM_CASE_CAPITAL, DM_CASE_UPPERCASE, DM_CASE_LOWERCASE)

imm_expand
Expand immediately (always TRUE for templates).

numfield
Number of fields (only valid for templates).

fields
Template fields.

affect
TRUE for macros that should be affected by macros (such as "italian 1").

DM_TEMPLATE_FIELD

Structure to hold the dictation macro template field information.

Template field structure

```
typedef struct
{
ULONG        type;
PCHAR        prompt;
PCHAR        defaulttext;
ULONG        position;
ULONG        numsel;
PCHAR        *selword;
} DM_TEMPLATE_FIELD, *PDM_TEMPLATE_FIELD;
```

Fields

type
Type of field (DM_FLDTYP_TEXT, DM_FLDTYP_SPELL, DM_FLDTYP_DIGIT, DM_FLDTYP_SELECTION).

prompt
Prompt text for field.

defaulttext
Default text of the field.

position
Position within text.

numsel
Number of selection words if type=DM_FLDTYP_SELECTION.

selword
Field of selection words.

Chapter 4

DMAPI Return Codes and Messages

These return codes and messages, defined in DMAPI.H, are generated by DMAPI.

DMAPI Return Codes and Messages

The following list contains the return code values in numeric order for DMAPI. Return codes marked with an asterisk (*) are used internally by the Dictation Macro Editor and are not returned to the application.

0	"DM_ERR_OK"	
1	"DM_ERR_TEMPLATE_NOT_FOUND"	
2	"DM_ERR_MACRO_NOT_FOUND"	
3	"DM_ERR_MALLOC"	
4	"DM_ERR_INVALID_USERID"	
5	"DM_ERR_INVALID_DOMAIN"	
6	"DM_ERR_INVALID_LANGUAGE"	
7	"DM_ERR_INVALID_HANDLE"	
8	"DM_ERR_INVALID_MACRO"	
9	"DM_ERR_INVALID_TEMPLATE"	*
10	"DM_ERR_DUPLICATE_MACRO"	
11	"DM_ERR_FILE_READ"	*
12	"DM_ERR_NOTIFY_EXCEEDED"	
13	"DM_ERR_OPEN_CLIPBOARD_FAILED"	*
14	"DM_ERR_NOT_ENOUGH_SHARED_MEMORY"	*
15	"DM_ERR_SHARED_MEM_UNDEFINED"	*
16	"DM_ERR_TRANSFER_TO_CLIPBRD"	*
17	"DM_ERR_MISSING_CLIPBRD_FORMAT"	*
18	"DM_ERR_INVALID_MACROFILE"	
19	"DM_ERR_INVALID_MACROVERSION"	*
20	"DM_ERR_INVALID_MACRORELEASE"	*
21	"DM_ERR_MACRO_NESTING"	*
22	"DM_ERR_INVALID_KEYWORD"	
23	"DM_ERR_INVALID_KEYTYPE"	
24	"DM_ERR_INVALID_KEYWORD"	
25	"DM_ERR_EMFILE"	
26	"DM_ERR_ENOENT"	

```
29          "DM_ERR_UNDEFINED"
30          "DM_ERR_LOAD_RESOURCE_DLL"
31          "DM_ERR_EBADF"
32          "DM_ERR_ENOSPC"                                 *
33          "DM_ERR_ACT_FILE_PARSE"                         *
34          "DM_ERR_INI_FILE_PARSE"                         *
35          "DM_ERR_EXPDLL_TIMEOUT"
36          "DM_ERR_EXPDLL_QUERYFUN_FAILED"
37          "DM_ERR_EXPDLL_LOAD_FAILED"
38          "DM_ERR_EXPDLL_ERR"
39          "DM_ERR_SYSTEM_PARM_LONG"
40          "DM_ERR_OUT_OF_SYSTEM_RES"
41          "DM_ERR_BAD_EXE_FORMAT"
42          "DM_ERR_NOUPDATE"
43          "DM_ERR_FILE_SET_AFFECT_FLAG"                   *
44          "DM_ERR_UNEXPECTED_EOF"                         *
45          "DM_ERR_MACRO_IS_INCLUDED"                      *
46          "DM_ERR_ACCESS_DENIED"
47          "DM_ERR_MACRONAME"
```

DMAPI Message Explanations

The following are explanations for the DMAPI messages in alphabetical order.

DM_ERR_ACCESS_DENIED

Explanation:

Access for modifying the current macro set is denied.

User response:

Try the call again with a different macro set.

DM_ERR_ACT_FILE_PARSE

Explanation:

This error code is used internally by the Dictation Macro Editor.

User response:

None.

DM_ERR_BAD_EXE_FORMAT

Explanation:

The specified EXE file is invalid. It may be a non-Win32 EXE, or there may be an error in the EXE image.

User response:

Correct the application programming error.

DM_ERR_DUPLICATE_MACRO

Explanation:

An attempt was made to load an already existing macro.

User response:

None.

DM_ERR_EACCESS

Explanation:

An error occurred attempting to access a macro file. Either the specified file is a directory, or you attempted to write to a read-only file.

User response:

Correct the file name and try the call again.

DM_ERR_EBADF

Explanation:

The file is not opened.

User response:

Correct the application programming error.

DM_ERR_EMFILE

Explanation:

No more file handles are available.

User response:

Correct the application programming error.

DM_ERR_ENOENT

Explanation:

The specified file or path was not found.

User response:

Try again with a different file or path name.

DM_ERR_ENOSPC

Explanation:

This error code is used internally by the Dictation Macro Editor.

User response:

None.

DM_ERR_EXPDLL_ERR

Explanation:

An error occurred executing a DLL function.

User response:

Correct the application programming error.

DM_ERR_EXPDLL_LOAD_FAILED

Explanation:

DMAPI was unable to load a DLL.

User response:

Correct the application programming error.

DM_ERR_EXPDLL_QUERYFUN_FAILED

Explanation:

DMAPI was unable to load a function of a DLL.

User response:

Correct the application programming error.

DM_ERR_EXPDLL_TIMEOUT

Explanation:

The execution of a DLL function timed out.

User response:

Make sure that the call does not take longer than two seconds.

DM_ERR_FILE_READ

Explanation:

This error code is used internally by the Dictation Macro Editor.

User response:

None.

DM_ERR_INI_FILE_PARSE

Explanation:

This error code is used internally by the Dictation Macro Editor.

User response:

None.

DM_ERR_INVALID_DOMAIN

Explanation:

An invalid domain name was passed to DMAPI.

User response:

Correct the domain name and try the call again.

DM_ERR_INVALID_HANDLE

Explanation:

An invalid macro handle was passed to DMAPI.

User response:

Correct the macro handle and try the call again.

DM_ERR_INVALID_KEYTYPE

Explanation:

An invalid key type was passed to DMAPI.

User response:

Correct the type and try the call again.

DM_ERR_INVALID_KEYWORD

Explanation:

An invalid keyword was passed to DMAPI.

User response:

Correct the keyword and try the call again.

DM_ERR_INVALID_LANGUAGE

Explanation:

An invalid language code was passed to DMAPI.

User response:

Correct the language code and try the call again.

DM_ERR_INVALID_MACRO

Explanation:

An attempt was made to load an invalid macro. The macro file may be corrupted.

User response:

None.

DM_ERR_INVALID_MACROFILE

Explanation:

The macro file specified is invalid.

User response:

Correct the macro file name and try again.

DM_ERR_INVALID_MACRORELEASE

Explanation:

This error code is used internally by the Dictation Macro Editor.

User response:

None.

DM_ERR_INVALID_MACROVERSION

Explanation:

This error code is used internally by the Dictation Macro Editor.

User response:

None.

DM_ERR_INVALID_TEMPLATE

Explanation:

This error code is used internally by the Dictation Macro Editor.

User response:

None.

DM_ERR_INVALID_USERID

Explanation:

An invalid user ID was passed to DMAPI.

User response:

Correct the user ID and try the call again.

DM_ERR_LOAD_RESOURCE_DLL

Explanation:

DMAPI was unable to load the specified DLL.

User response:

Correct the application programming error.

DM_ERR_MACRO_NESTING

Explanation:

This error code is used internally by the Dictation Macro Editor.

User response:

None.

DM_ERR_MACRO_IS_INCLUDED

Explanation:

This error code is used internally by the Dictation Macro Editor.

User response:

None.

DM_ERR_MACRONAME

Explanation:

Invalid macroname (empty or too long).

User response:

Try the call again with a different macro name.

DM_ERR_MACRO_NOT_FOUND

Explanation:

DMAPI was unable to find the specified macro. Either a macro does not exist by that name, or you specified an incorrect name.

User response:

Try the call again with a different macro name.

DM_ERR_MALLOC

Explanation:

A memory allocation failure occurred.

User response:

Correct the application programming error.

DM_ERR_MISSING_CLIPBRD_FORMAT

Explanation:

This error code is used internally by the Dictation Macro Editor.

User response:

None.

DM_ERR_NOT_ENOUGH_SHARED_MEMORY

Explanation:

This error code is used internally by the Dictation Macro Editor.

User response:

None.

DM_ERR_NOTIFY_EXCEEDED

Explanation:

The maximum number of Dictation Macro Editor sessions has been exceeded.

User response:

Close a Dictation Macro Editor session and try the call again.

DM_ERR_NOUPDATE

Explanation:

This error is returned by DmUpdate when no changes have been made to any macros or templates since the last DmUpdate call.

User response:

None.

DM_ERR_OK

Explanation:

No error occurred.

User response:

None.

DM_ERR_OPEN_CLIPBOARD_FAILED

Explanation:

This error code is used internally by the Dictation Macro Editor.

User response:

None.

DM_ERR_OUT_OF_SYSTEM_RES

Explanation:

The system is out of memory or other system resources.

User response:

Correct the application programming error.

DM_ERR_FILE_SET_AFFECT_FLAG

Explanation:

This error code is used internally by the Dictation Macro Editor.

User response:

None.

DM_ERR_SHARED_MEM_UNDEFINED

Explanation:

This error code is used internally by the Dictation Macro Editor.

User response:

None.

DM_ERR_SYSTEM_PARM_LONG

Explanation:

A parameter specified in a system call is too long.

User response:

Correct the error and try the call again.

DM_ERR_TEMPLATE_NOT_FOUND

Explanation:

DMAPI was unable to find the specified template. Either a template does not exist by that name, or you specified an incorrect name.

User response:

Try the call again with a different template name.

DM_ERR_TRANSFER_TO_CLIPBRD

Explanation:

This error code is used internally by the Dictation Macro Editor.

User response:

None.

DM_ERR_UNDEFINED

Explanation:

An undefined file access error occurred.

User response:

Correct the application programming error.

DM_ERR_UNEXPECTED_EOF

Explanation:

This error code is used internally by the Dictation Macro Editor.

User response:

None.

Appendix A Notices

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program or service is not intended to state or imply that only that IBM product, program, or service may be used.

Subject to IBM's valid intellectual property or other legally protectable rights, any functionally equivalent product, program, or service may be used instead of the IBM product, program, or service.

The evaluation and verification of operation in conjunction with other products, except those expressly designated by IBM, are the responsibility of the user.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armand, NY 10504-1785
USA

Asia-Pacific users can inquire, in writing, to the IBM Director of Intellectual Property and Licensing, IBM World Trade Asia Corporation, 2-31 Roppongi 3-chome, Minato-ku, Tokyo 106, Japan.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact IBM Corporation, Department T01B, 3039 Cornwallis, Research Triangle Park, NC 27709-2195, USA. Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

Trademarks

The following terms are trademarks of the IBM Corporation in the United States or other countries or both:

IBM

ViaVoice

VoiceType

Adobe Acrobat is a trademark or registered trademark of Adobe Systems Incorporated.

Intel and Pentium are trademarks or registered trademarks of Intel Corporation in the United States and/or other countries.

Microsoft, Windows, Windows NT, and the Windows 95 logo are trademarks or registered trademarks of Microsoft Corporation in the United States and/or other countries.

Other company, product, and service names may be trademarks or service marks of others.

Index