Office APIs (V5R2)

Table of Contents

Office APIs

The office APIs work with system distribution directory data and with document handling. These APIs allow you to perform customized and additional actions with office data.

The office APIs consist of:

- Office-related APIs
- AnyMail/400 Mail Server Framework APIs
- SNADS File Server Object APIs

For additional information, see Word Separator Tables.

APIs by category

Office-related APIs

The office-related APIs allow you to perform customized and additional actions with office data.

The office APIs are:

- Add Department (QOKADDDP) allows you to add a department to the system distribution directory.
- Aid Spelling (QTWAIDSP) allows you to retrieve a list of correctly spelled words that are similar in spelling to the input word.
- Change Department (QOKCHGDP) allows you to change a department definition in the system distribution directory.
- Change Office Program (QOGCHGOE) allows you to set or change the document handling and document conversion exit programs.
- Check Spelling (QTWCHKSP) accepts a list of one or more words and returns the list with an indicator of whether each word is valid.
- Control Office Services (QOCCTLOF) makes requests of office services and indicates that several office tasks will be processed.
- Display Directory Panels (QOKDSPDP) allows you to use the Change Directory Information display interactively without using OfficeVision administration to change directory information for OfficeVision users.
- Display Directory X.400 Panels (QOKDSPX4) llows you to display directory X.400 O/R name panels and process functions interactively without going through the Work with Directory panel.
- Remove Department (QOKRMVDP) allows you to remove a department from the system distribution directory.
- Retrieve Office Programs (QOGRTVOE) allows you to retrieve program names and attributes for the current document handling and document conversion exit programs.
- Search System Directory (QOKSCHD) allows you to search the system distribution directory and receive results that match the input criteria.

The office exit programs are:

- Directory Maintenance allows the administrator to define additional security or validation checking when directory data is added, changed, or deleted. There are two entry points. The verification entry is called before an addition, change, or deletion. The notification entry is called after the actions. The verification maintenance exit program is specified on the VRFPGM parameter of the Change System Directory Attribute (CHGSYSDIRA) command. The verification maintenance exit program can also be specified with the Work with Registration Information (WRKREGINF) command. The notification maintenance exit program is specified with the Work with Registration Information (WRKREGINF) command.
- Directory Search allows the administrator to define a customized search of directory data. The Directory Search program is specified on the SCHPGM parameter of the Change System Directory Attribute (CHGSYSDIRA) command.
- Directory Supplier allows the administrator to decide whether a change, add, or delete operation for directory entries, departments, and locations is to be shadowed to collector systems. The supplier program is specified on the SUPPGM parameter of the Change System Directory Attribute (CHGSYSDIRA) command.
- Document Conversion allows other document conversion programs to be called when a request is made for the OfficeVision program to process a document that is an unsupported type (for example, PCFILE).

- Document Handling allows other editors and applications to be called from the OfficeVision word processing and print functions.
- User Application Administration passes control to the application enabler where a registered alternate administration program will be called.

Top | Office APIs | APIs by category

Add Department (QOKADDDP) API

Required Parameter Group:

1	Department	Input	Char(10)
2	Title	Input	Char(50)
3	Manager	Input	Char(16)
4	Reports to department	Input	Char(10)
5	Allow duplicate departments	Input	Char(1)
6	Error code	I/O	Char(*)

Default Public Authority: *USE

Threadsafe: No

The Add Department (QOKADDDP) API adds departments to the system distribution directory. To add and remove members from within a department use the DEPT parameter on the Add Directory Entry (ADDDIRE) and Change Directory Entry (CHGDIRE) CL commands.

Authorities and Locks

Adding a department requires *SECADM (security administrator) authority.

Required Parameter Group

Department

INPUT; CHAR(10)

The department name to add. The department name must not be blank.

Title

INPUT; CHAR(50)

The descriptive title of the department. The title can be left blank.

Manager

INPUT; CHAR(16)

The department manager's user ID and address. The first 8 characters contain the user ID and the last 8 contain the address. The manager user ID and address must exist in the system distribution directory if specified. The manager field can be left blank.

Reports to department

INPUT; CHAR(10)

The department to which this department reports. If a department that does not exist in the system distribution directory is specified, it is automatically created. This field can be left blank.

Allow duplicate departments

INPUT; CHAR(1)

Whether to allow duplicate department names. The valid values are:

0	Do not allow duplicate department names.
1	Allow duplicate department names.

The department name value is not case sensitive when checking for duplicate departments.

Error code

I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3C90 E	Literal value cannot be changed.
CPF3CF1 E	Error code parameter not valid.
CPF89A3 E	Operation not successful due to authority reasons.
CPF89A4 E	Operation not successful due to data validation reasons.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.
CPF9A8D E	Error occurred with &5 API. Reason code is &1.

API Introduced: V3R6

Aid Spelling (QTWAIDSP) API

Required Parameter Group:

1	Receiver variable	Output	Char(*)
2	Length of receiver variable	Input	Binary(4)
3	Format name	Input	Char(8)
4	Input word	Input	Char(*)
5	Length of input word	Input	Binary(4)
6	Input dictionaries	Input	Char(*)
7	Length of input dictionaries	Input	Binary(4)
8	Output dictionaries	Output	Char(*)
9	Length of output dictionaries	Input	Binary(4)
10	Error code	I/O	Char(*)

Default Public Authority: *USE

Threadsafe: No

The Aid Spelling (QTWAIDSP) API returns a list of candidate words that are similar in spelling to the input word. The input word is generally the misspelled word and the candidate words are correctly spelled words that are spelled similarly to the input word. The QTWAIDSP API returns a maximum of six candidate words per dictionary.

The QTWAIDSP API can be used to do the following:

- Return an indication if the input word was misspelled
- Return a list of correctly spelled words which are similar to the input word

To perform the function, a user can specify a maximum of eight IBM or user language dictionaries in the input dictionaries parameter.

Authorities and Locks

Dictionary Authority

*USE

Dictionary Library Authority

*USE

Dictionary Lock Authority

*SHRNUP

Required Parameter Group

Receiver variable

OUTPUT; CHAR(*)

The variable that is to receive the information about candidate words. You can specify the size of this area to be smaller than necessary to hold information about each word as long as you specify the length parameter correctly. The API returns only the data that the area can hold. The minimum size area is 8 bytes.

Length of receiver variable

INPUT; BINARY(4)

The length of the receiver variable. If the length is larger than the size of the receiver variable, the results may not be predictable. The minimum length is 8 bytes.

Format name

INPUT; CHAR(8)

The content and format of the information returned in the receiver variable. You can use this format:

AIDW0100	Return all candidate words. See AIDW0100 Format.

Input word

INPUT; CHAR(*)

Locate candidate words that are spelled similarly to this word.

Length of input word

INPUT; BINARY(4)

The length of the input word.

Input dictionaries

INPUT; CHAR(*)

A structure containing a list of up to eight dictionaries that is used to locate candidate words.

Length of input dictionaries

INPUT; BINARY(4)

The length of the input dictionaries parameter. The only valid value is 172 bytes.

Output dictionaries

OUTPUT; CHAR(*)

A structure containing a list of up to eight spelling aid dictionaries that were actually used to find candidate words.

Length of output dictionaries

INPUT; BINARY(4)

The length of the output dictionaries parameter. The following are valid values:

0	No dictionaries are returned in the output dictionaries parameter.

>0 The length of space available in the output dictionaries parameter. If this length is larger than the actual size of the output dictionaries parameter, the results may not be predictable.

Error code

I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter.

AIDW0100 Format

The following information is returned in the receiver variable for the AIDW0100 format. For detailed descriptions of the fields, see Field Descriptions.

Offset			
Dec	**Hex**	**Type**	**Field**
Note: Fixed section.			
0	0	BINARY(4)	Bytes returned
4	4	BINARY(4)	Bytes available
8	8	BINARY(4)	Number of words returned
12	C	BINARY(4)	Number of words available
16	10	BINARY(4)	Offset to input word
20	14	BINARY(4)	Length of input word
24	18	CHAR(1)	Misspelled
25	19	CHAR(3)	Reserved
28	1C	BINARY(4)	Offset to first word information entry
32	20	BINARY(4)	Length of word information entry
36	24	BINARY(4)	Reserved
Note: The input word for which candidate words were found. The decimal and hexadecimal offsets are not defined. The input word is found by using the offset to input word field.			
		CHAR(*)	Input word
Note: Format of a word information entry. The following fields are repeated for each word returned. The decimal and hexadecimal offsets depend on the number of words returned. The first word information entry is found by using the offset to first word information entry in the fixed section. Each subsequent word information entry is found by adding the length of a word information entry to the offset of the previous word information entry.			
		BINARY(4)	Offset to candidate word
		BINARY(4)	Length of candidate word
		BINARY(4)	Output dictionary number
		CHAR(*)	Reserved
Note: The following field is repeated for each word returned. Each word is located by the offset to candidate word field and the length of the word is specified in the length of candidate word field.			

		CHAR(*)	Candidate word

Field Descriptions

Bytes available. The total length of all data available.

Bytes returned. The length of the data actually returned. If the data is truncated because the receiver variable is not large enough to hold all of the data available, the value will be less than the bytes available.

Candidate word. The candidate word that is spelled similar to the input word.

Input word. The word actually used to find candidate words.

Length of candidate word. The length of the candidate word in bytes.

Length of input word. The length of the input word in bytes.

Length of word information entry. The length of one item in the word information entry section.

Misspelled. Determines if the word is misspelled.

0	The word is spelled correctly.
1	The word is misspelled.

Number of words available. The number of candidate words that were found.

Number of words returned. The number of words actually returned in the receiver variable. If this number is smaller than the number of words available, then all of the words could not fit in the receiver variable.

Offset to candidate word. The byte offset, from the beginning of the receiver variable to the candidate word.

Offset to input word. The byte offset, from the beginning of the receiver variable to the input word.

Offset to first word information entry. The byte offset, from the beginning of the receiver variable to the first item in the word information entry section. If no candidate words are found, the value will be 0.

Output dictionary number. The index into the output dictionary list where the candidate word originated.

Reserved. An ignored field.

Error Messages

Message ID	**Error Message Text**
CPF24B4 E	Severe error while addressing parameter list.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPF3C19 E	Error occurred with receiver variable specified.
CPF3C21 E	Format name &1 is not valid.

CPF3C24 E	Length of the receiver variable is not valid.
CPF3C90 E	Literal value cannot be changed.
CPF8751 E	Number of dictionaries is not valid.
CPF8752 E	No valid dictionaries were found.
CPF8754 E	Length of input word not valid.
CPF8755 E	Length of input dictionaries not valid.
CPF8756 E	Length of output dictionaries not valid.
CPF8757 E	Input word &1 is not valid.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API Introduced: V3R1

Top | Office APIs | APIs by category

Change Department (QOKCHGDP) API

Required Parameter Group:

1	Department	Input	Char(10)
2	Title	Input	Char(50)
3	Manager	Input	Char(16)
4	Reports to department	Input	Char(10)
5	New department name	Input	Char(10)
6	Allow duplicate departments	Input	Char(1)
7	Error code	I/O	Char(*)

Default Public Authority: *USE

Threadsafe: No

The Change Department (QOKCHGDP) API changes department information in the system distribution directory. To add and remove members from within a department use the DEPT parameter on the Add Directory Entry (ADDDIRE) and Change Directory Entry (CHGDIRE) CL commands.

Authorities and Locks

Changing a department requires *SECADM (security administrator) authority.

Required Parameter Group

Department

INPUT; CHAR(10)

The department name to change. The department name must not be blank. When there are duplicate department names in the directory, the first match found is changed. The department name value is not case sensitive when being checked.

Title

INPUT; CHAR(50)

The descriptive title of the department. If this field is to remain unchanged, you must use *SAME for the field value. Blanks change the field to blanks.

Manager

INPUT; CHAR(16)

The department manager's user ID and address. The first 8 characters contain the user ID and the last 8 contain the address. The manager user ID and address must exist in the system distribution directory if specified. If this field is to remain unchanged, you must use *SAME for the field value. Blanks change the field to blanks.

Reports to department

INPUT; CHAR(10)

The department to which this department reports. If a department that does not exist in the system distribution directory is specified, it is automatically created. If this field is to remain unchanged, you must use *SAME for the field value. Blanks change the field to blanks. The department name value is not case sensitive when being checked.

New department.

INPUT; CHAR(10)

The new department name. If the department name is not to be changed, you must use *SAME for the field value. Blanks are not allowed. If this field is not *SAME, the department name is changed to this value. This changes the department value in all the system distribution directory entries that are members of the department being renamed.

Allow duplicate departments

INPUT; CHAR(1)

Whether to allow duplicate department names. The valid values are:

0	Do not allow duplicate department names.
1	Allow duplicate department names.

The department name value is not case sensitive when checking for duplicate departments.

Error code

I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3C90 E	Literal value cannot be changed.
CPF3CF1 E	Error code parameter not valid.
CPF89A3 E	Operation not successful due to authority reasons.
CPF89A4 E	Operation not successful due to data validation reasons.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.
CPF9A8D E	Error occurred with &5 API. Reason code is &1.

API Introduced: V3R6

Change Office Program (QOGCHGOE) API

Required Parameter Group:

1	Qualified document handling exit program	Input	Char(20)
2	Document handling program supports mail flag	Input	Char(1)
3	Qualified document conversion exit program	Input	Char(20)
4	Error code	I/O	Char(*)

Optional Parameter Group:

5	Activate application enabler support	Input	Char(1)
6	Activate mail handling support	Input	Char(1)
7	Activate PCFILE identification processing support	Input	Char(1)

Default Public Authority: *USE

Threadsafe: No

The Change Office Program (QOGCHGOE) API allows you to set or change the document handling and document conversion exit programs. The default value for these exit programs is *IBM, which indicates that OfficeVision processing should be used for the request or conversion.

Authority

You must have security administrator (*SECADM) authority to call this program.

Required Parameter Group

Qualified document handling exit program

INPUT; CHAR(20)

The name of the program called when document editing and print requests are made. The first 10 characters contain the program name, and the second 10 characters contain the name of the library where the program is located.

**IBM*	OfficeVision should process document requests. The library name should be blanks if *IBM is specified in the program name.

Document handling program supports mail flag

INPUT; CHAR(1)

This flag indicates whether the document handling program supports mail requests (that is, MAILVIEW, MAILEDIT, EDIT using Revise a Copy, MAILFWD, MAILREPLY, PRINT and PRINTOPTS from the Work with Mail display). If an application does not plan to handle these functions, set the value to 0. Setting the value to 0 prevents the unnecessary overhead of filing the mail items and calling the document handling program. If the document handling program is *IBM, set the value to 1.

0	Mail requests are not supported.
1	Mail requests are supported.

This indicates whether the program specified in the document handling exit program and library name parameter should be called for mail requests.

Qualified document conversion exit program

INPUT; CHAR(20)

The name of the program that is used to do document conversions when office services requires the document to be in a different data stream format. The first 10 characters contain the program name, and the second 10 characters contain the name of the library where the program is located.

**IBM*	Use the IBM-supplied document conversion exit program. The library name should be blank if *IBM is specified in the program name.

Error code

I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter.

Optional Parameter Group

Activate application enabler support

INPUT; CHAR(1)

If an application does not plan to handle these functions, set the value to 0. If the application requires application enabler support, set the value to 1. Any other value does not change the current activation of application enabler support.

0	Application enabler support is inactive.
1	Application enabler support is active.

Activate mail handling support

INPUT; CHAR(1)

If an application does not plan to handle these functions, set the value to 0. If the program requires mail handling support, set the value to 1. Any other value does not change the current activation of mail handling support.

0	Mail handling support is inactive.
1	Mail handling support is active.

Activate PCFILE identification processing support

INPUT; CHAR(1)

If an application does not plan to handle these functions, set the value to 0. If the program requires PCFILE identification processing support, (1) set the value to 1. Any other value does not change the current activation of PCFILE identification processing support.

0	PCFILE identification processing support is inactive.
1	PCFILE identification processing support is active.

Error Messages

Message ID	Error Message Text
CPF3C90 E	Literal value cannot be changed.
CPF3C29 E	Object name &1 is not valid.
CPF90A8 E	*SECADM special authority required to do requested operation.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.
OFC15FF E	Value &1 for mail handled parameter is not valid.
OFC8019 E	Required module not on system.

API Introduced: V2R2

Check Spelling (QTWCHKSP) API

Required Parameter Group:

1	Receiver variable	Output	Char(*)
2	Length of receiver variable	Input	Binary(4)
3	Format name	Input	Char(8)
4	Word list	Input	Char(*)
5	Length of word list	Input	Binary(4)
6	Input dictionaries	Input	Char(*)
7	Length of input dictionaries	Input	Binary(4)
8	Output dictionaries	Output	Char(*)
9	Length of output dictionaries	Input	Binary(4)
10	Error code	I/O	Char(*)

Default Public Authority: *USE

Threadsafe: No

The Check Spelling (QTWCHKSP) API accepts a list of one or more words in the word list parameter, and returns one word array entry in the receiver variable for every word with an indication if it is a valid word. To retrieve additional information about a specific word, see the Aid Spelling (QTWAIDSP) API.

The QTWCHKSP API can be used to do the following:

- Check the spelling of an individual word
- Find misspelled words in a character string

To perform the function, a user can specify a maximum of eight IBM or user language dictionaries in the input dictionaries parameter.

Authorities and Locks

Dictionary Authority

*USE

Dictionary Library Authority

*USE

Dictionary Lock Authority

*SHRNUP

Required Parameter Group

Receiver variable

OUTPUT; CHAR(*)

The variable that is to receive the information about each word in the word list. You can specify the size of this area to be smaller than necessary to hold information about each word as long as you specify the length parameter correctly. As a result, the API returns only the data that the area can hold. The minimum size of this area is eight bytes.

Length of receiver variable

INPUT; BINARY(4)

The length of the receiver variable. If the length is larger than the size of the receiver variable, the results may not be predictable. The minimum length is 8 bytes.

Format name

INPUT; CHAR(8)

The content and format of the information returned in the receiver variable. For more information on the format see CHKW0100 and CHKW0200 Formats. The following are possible format names:

CHKW0100	Word array entries for only the misspelled words should be returned in the receiver variable.
CHKW0200	Word array entries for all words in the word list should be returned.

Word list

INPUT; CHAR(*)

A list of one or more words to be checked. The words must be separated by one or more word separators where the blank character is always considered a word separator. The CCSID of the current job determines what code page will be used and each code page defines what other characters can be used as word separators. Refer to the Globalization topic in the iSeries Information Center for more information on CCSIDs. Refer to Word Separator Tables for information on word separators.

Length of word list

INPUT; BINARY(4)

The length of the word list in bytes.

Input dictionaries

INPUT; CHAR(*)

A structure containing a list of up to eight dictionaries to be used to determine if the word or words in the word list are spelled correctly. For the format of this parameter, see Input Dictionaries Format.

Length of input dictionaries

INPUT; BINARY(4)

The length of the input dictionaries parameter. The only valid value is 172 bytes.

Output dictionaries

OUTPUT; CHAR(*)

A structure containing a list of up to eight dictionaries that were actually used. For the format of this parameter, see Output Dictionaries Format.

Length of output dictionaries

INPUT; BINARY(4)

The length of the output dictionaries parameter. The following are valid values:

0	No dictionaries are returned in the output dictionaries parameter.
>0	The length of space available in the output dictionaries parameter. If this length is larger than the actual size of the output dictionaries parameter, the results may not be predictable.

Error code

I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter.

CHKW0100 and CHKW0200 Formats

The following information is returned for both the CHKW0100 and CHKW0200 formats. If the receiver variable is large enough to hold all the information, then it contains information about each word in the word list.

The receiver variable has three logical sections which contain:

- Fixed section
- Word information entry
- An array of the actual input words

For more detailed descriptions of the fields in the table, see Field Descriptions.

Offset			
Dec	**Hex**	**Type**	**Field**
Note: Fixed section.			
0	0	BINARY(4)	Bytes returned
4	4	BINARY(4)	Bytes available
8	8	BINARY(4)	Number of words returned
12	C	BINARY(4)	Number of words available
16	10	BINARY(4)	Offset to first word information entry
20	14	BINARY(4)	Length of word information entry
24	18	BINARY(4)	Reserved
Note: Format of a word information entry. The following four fields are repeated for each word returned. The first word information entry is found by using the offset to first word information entry in the fixed section. Each subsequent word information entry is found by adding the length of a word information entry to the offset of the previous word information entry.			
		BINARY(4)	Offset to word

		BINARY(4)	Length of word
		CHAR(1)	Misspelled
		CHAR(*)	Reserved
Note: The following field is repeated for each word returned. Each word is located by the offset to word field and the length of the word is specified in the length of word field.			
		CHAR(*)	Word

Input Dictionaries Format

The following list shows the format of the input dictionaries parameter. For detailed descriptions of the fields, see Field Descriptions.

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	BINARY(4)	Offset to dictionary entries
4	4	BINARY(4)	Number of dictionary entries
8	8	BINARY(4)	Reserved (must be set to 0)
Note: Format of a dictionary entry. The following fields are repeated by the number of dictionary entries. The decimal and hexadecimal offsets depend on the number of dictionary entries. The first dictionary entry is found by using the offset to dictionary entries in the header section.			
		CHAR(10)	Dictionary name
		CHAR(10)	Dictionary library name

Output Dictionaries Format

The following shows the format of the output dictionaries parameter. For detailed descriptions of the fields, see Field Descriptions.

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	BINARY(4)	Number of dictionaries returned
4	4	BINARY(4)	Number of dictionaries available
Note: The following fields are repeated by the number of dictionaries returned. The decimal and hexadecimal offsets depend on the number of dictionaries returned.			
		CHAR(10)	Dictionary name
		CHAR(10)	Dictionary library name

Field Descriptions

Bytes available. The total length of all data available.

Bytes returned. The length of the data actually returned. If the receiver variable is not large enough to hold all of the data available, only the data that will fit in the space available is returned and this value will be less that the bytes available.

Dictionary library name. The library containing the language dictionary. The special values for the library name are *CURLIB and *LIBL.

Dictionary name. The name of the language dictionary.One special value for the name is *USERID.

**USERID*	The name of the dictionary is the same as the user ID for the current job. A library name must be specified with this value.

Length of word. The length of the word in bytes.

Length of word information entry. The length of one item in the word information entry section.

Misspelled. Indicates if the word is misspelled.

0	The word is spelled correctly.
1	The word is misspelled.

Number of dictionaries available. The number of dictionaries that were actually used to determine if the word or words in the word list were spelled correctly.

Number of dictionaries returned. The number of dictionaries actually returned in the output dictionaries parameter. If this number is smaller than the number of dictionaries available, then all of the dictionaries could not fit in the space available.

Number of dictionary entries. The number of usable dictionaries to perform the function. Valid values are 1 through 8.

Number of words available. The number of words which are found in the word list.

Number of words returned. The number of words actually returned in the receiver variable. If this number is smaller than the number of words available, then all of the words could not fit in the receiver variable.

Offset to dictionary entries. The byte offset, from the beginning of the parameter, to the beginning of the dictionary entries.

Offset to first word information entry. The byte offset, from the beginning of the receiver variable, to the first item in the word information entry section.

Offset to word. The byte offset, from the beginning of the receiver variable, to the actual word.

Reserved. This field must be set to 0 on the input dictionaries parameter. Otherwise, this field is ignored.

Word. The word that was checked for spelling.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPF3C19 E	Error occurred with receiver variable specified.
CPF3C21 E	Format name &1 is not valid.
CPF3C24 E	Length of the receiver variable is not valid.
CPF3C90 E	Literal value cannot be changed.
CPF8751 E	Number of dictionaries is not valid.
CPF8752 E	No valid dictionaries were found.
CPF8753 E	Length of word list not valid.
CPF8755 E	Length of input dictionaries not valid.
CPF8756 E	Length of output dictionaries not valid.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API Introduced: V2R3

Top | Office APIs | APIs by category

Control Office Services (QOCCTLOF) API

Required Parameter Group:

1	Request type	Input	Char(10)
2	Error code	I/O	Char(*)

Default Public Authority: *USE

Threadsafe: No

The Control Office Services (QOCCTLOF) API makes requests of the office services. Office services accepts the following actions:

- *START
- *END
- *CHECK

An office services block is defined as a time during an application that the application knows that it will be using more office commands. By starting an office services block, the application is informing office that it will do several office tasks. The application will also inform office when to close the office files, destroy the working spaces, and so forth. It closes the files by ending an office services block. The application uses the QOCCTLOF to start and end an office services block.

For example, assume you have a program that issues 10 calendar CL commands. Each CL command must open office files and create work spaces that it needs before processing the request. The CL command must then close the office files that were opened and destroy the work spaces created before exiting. In this case, the files would be opened and closed and the work spaces created and destroyed 10 times, once for each CL command. With the QOCCTLOF API, you can improve the performance of this program by starting an office services block at the beginning of the program, issuing the CL commands and then ending the office services block. This enables the files to be opened and closed and the work spaces to be created and destroyed only once.

With the QOCCTLOF API, you can improve the performance of this program as follows:

- Use the QOCCTLOF API to start an office services block
- Issue the 10 calendar CL commands
- Use the QOCCTLOF API to end the office services block

All calendar CL commands can make use of this API.

Required Parameter Group

Request type

INPUT; CHAR(10)

The request that the application makes of the office services. The following are the possible values:

**START*	Start an office services block. An office services block is the time during the running of an application when more office commands will be used. Office services leaves the job in such a state as to maximize the performance of subsequent office services use. This includes leaving the office files open and leaving working spaces created and already initialized. It is the responsibility of the application to end the office services block.
**END*	End an office services block. No further office services will be used. The files will be closed and all working spaces deleted.
**CHECK*	Determine if an office services block is active. An error will be returned if an office services block has not been previously started or has already ended.

Error code

I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter.

Error Messages

Message ID	Error Message Text
CPF3C90 E	Literal value cannot be changed.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.
OFCFD01 E	Office services session is already active.
OFCFD02 E	Office services session request failed.
OFCFD03 E	Office services session is not active.
OFCFD04 E	Office services session request is not correct.
OFCFD05 E	Number of bytes provided for the error code is not correct.

API Introduced: V2R2

Display Directory Panels (QOKDSPDP) API

Required Parameter Group:

1	User ID	Input	Char(8)
2	User address	Input	Char(8)
3	Title	Input	Char(10)
4	Function key processing	Output	Char(10)
5	Message to be displayed	Input	Char(*)
6	Error code	I/O	Char(*)

Default Public Authority: *USE

Threadsafe: No

The Display Directory Panels (QOKDSPDP) API has the ability to display the Add Directory Information and Change Directory Information displays. It can run that function interactively without using OfficeVision administration to change directory information for office users. This API is for interactive use only.

Authorities and Locks

If you have security administrator (*SECADM) authority, you can change all the directory information. If you do not have security administrator authority, then you can only change your own directory information.

Required Parameter Group

User ID

INPUT; CHAR(8)

The user ID to be processed. The ID needs to be in character set 697, code page 500, and monocase.

User address

INPUT; CHAR(8)

The address of the user to be processed. The address needs to be in character set 697, code page 500, and monocase.

Title

INPUT; CHAR(10)

The type of directory panel to be displayed. The following are the possible values:

**CHG*	Change directory information
**ADD*	Add directory information

Function key processing

OUTPUT; CHAR(10)

The operation processed for the key pressed. This indicates what type of ending processing to do. The following are the possible values:

**ENTER*	The enter operation is processed.
**F3*	The exit operation is processed.
**F12*	The previous display is shown.

Message to be displayed

INPUT; CHAR(*)

A structure with a message ID and replacement text that is to be shown on the display. If the message size is 0, no message will be displayed. For the format of the structure, see Message to Be Displayed Format.

Error code

I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter.

Message to Be Displayed Format

The format of the message to be displayed parameter is as follows:

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	BINARY(2)	Message size
2	2	CHAR(7)	Message ID
9	9	CHAR(8)	Message file
17	11	CHAR(4)	Message type
21	15	BINARY(2)	Length of message data
23	17	CHAR(*)	Message data

Field Descriptions

Length of message data. The length of the message data. The maximum length is 256.

Message data. The data to be replaced in the message. The maximum size is 256.

Message file. The message file name where the message is displayed from. When the message is sent, it picks up the message library from *LIBL.

Message ID. The message ID of the message to be displayed.

Message size. The total length of the message to be displayed format. If this value is greater than zero, the message will be sent.

Message type. The message type of the message to be signaled. The valid values are:

INFO	Informational
DIAG	Diagnostic
ESCP	Escape
COMP	Completion
NTFY	Notify
STAT	Status

Error Messages

Message ID	**Error Message Text**
CPF3C90 E	Literal value cannot be changed.
CPF89A0 E	Values passed to QOKDSPDP are not valid
CPF9024 E	System cannot get correct record to finish operation.
CPF9054 E	Description '&1' already exists.
CPF9083 E	User ID and address &1 &2 not changed.
CPF9810 E	Library &1 not found.
CPF9830 E	Cannot assign library &1.
CPF9845 E	Error occurred while opening file &1.
CPF9846 E	Error while processing file &1 in library &2.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API Introduced: V2R2

Display Directory X.400 Panels (QOKDSPX4) API

Required Parameter Group:

1	Function	Input	Char(10)
2	Format	Input	Char(10)
3	System name	Input	Char(8)
4	System group	Input	Char(8)
5	User ID	I/O	Char(8)
6	User address	I/O	Char(8)
7	Text description	Output	Char(50)
8	Key code	Output	Char(10)
9	Error code	I/O	Char(*)

Default Public Authority: *USE

Threadsafe: No

The Display Directory X.400 Panels (QOKDSPX4) API has the ability to display various X.400 O/R name panels and to process functions interactively without going through the Work with Directory panel. An O/R name is to X.400 as a user ID and address is to SNADS; that is, it is the way users are addressed in an X.400 network.

This API is for interactive use only; it cannot be used for batch programs.

Authorities and Locks

The QOKDSPX4 API is shipped with *EXCLUDE public authority. If the program calling this API adopts authority, it is the responsibility of the calling program to ensure that the user has the correct authority to add X.400 O/R names.

Required Parameter Group

Function

INPUT; CHAR(10)

The function requested. The following are possible values:

**ADD*	Add O/R name information
**DISPLAY*	Display O/R name information

Format

INPUT; CHAR(10)

This field must be blank.

System name

INPUT; CHAR(8)

The system name (REN) of the user to be associated with the O/R name to be added. The system name cannot be the local system name because local user profiles are not automatically created. If blanks are passed when FUNCTION(*ADD) is specified, the local system name will be used. This parameter is ignored when the function is (*DISPLAY).

System group

INPUT; CHAR(8)

The system group name (RGN) of the user to be associated with the O/R name to be added. This parameter is required for FUNCTION(*ADD). This parameter is required for FUNCTION(*DISPLAY) but is ignored.

User ID

I/O; CHAR(8)

The user ID (DEN) of the user associated with the function. For FUNCTION(*ADD), this is an output-only field. On the panel, the user ID (DEN) will be initialized to *GEN. The user may override this with a specified user ID, or a user ID will be generated. For *GEN, the generated user ID is the value returned in this field. Otherwise, the value in this field when Enter is pressed is the value returned. For FUNCTION(*DISPLAY), this is an input-only field.

User address

I/O; CHAR(8)

The user address (DGN) of the user associated with the function. For FUNCTION(*ADD), this is an output-only field. On the panel, the user address will be initialized to the system name value that is specified on this API. The user can change the user address. The value in this field when Enter is pressed is the value returned in this field. For FUNCTION(*DISPLAY), this is an input-only field.

Text description

OUTPUT; CHAR(50)

The description of the directory entry associated with the O/R name added. This parameter is valid only for FUNCTION(*ADD).

Key code

OUTPUT; CHAR(10)

The key code (F3, F12, Enter) from the X.400 panel. The following are possible values:

**F3*	F3 was pressed on the X.400 panel
**F12*	F12 was pressed on the X.400 panel
**ENTER*	Enter was pressed on the X.400 panel

Error code

I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter.

Error Messages

Message ID	Error Message Text
CPF3C90 E	Literal value cannot be changed.
CPF3CF1 E	Error code parameter not valid.
CPF89A3 E	Operation not successful due to authority reasons.
CPF89A4 E	Operation not successful due to data validation reasons.
CPF89A7 E	Error associated with exit program. Reason code &3. <</td>
CPF898A E	X.400 O/R name does not exist for this directory entry. </</td>
CPF9A8C E	Error occurred with QOKSCHD API. Reason code &1.
CPF9A82 E	Values passed to QOKDSPX4 not valid.
CPF9082 E	User ID and address &1 &2 not added to directory.
CPF9845 E	Error occurred while opening file &1.
CPF9846 E	Error while processing file &1 in library &2.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API Introduced: V3R1

Remove Department (QOKRMVDP) API

Required Parameter Group:

1	Department	Input	Char(10)
2	Error code	I/O	Char(*)

Default Public Authority: *USE

Threadsafe: No

The Remove Department (QOKRMVDP) API removes a department from the system distribution directory. The department field in the directory for the removed department members is blanked out. To remove individual members from within a department, use the DEPT parameter on the Add Directory Entry (ADDDIRE) and Change Directory Entry (CHGDIRE) CL commands.

Authorities and Locks

Removing a department requires *SECADM (security administrator) authority.

Required Parameter Group

Department

INPUT; CHAR(10)

The department name to remove. The department name must not be blank. When there are duplicate department names in the directory, the first match found is removed. The department name value is not case sensitive when being checked.

Error code

I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3C90 E	Literal value cannot be changed.
CPF3CF1 E	Error code parameter not valid.

CPF89A3 E	Operation not successful due to authority reasons.
CPF89A4 E	Operation not successful due to data validation reasons.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.
CPF9A8D E	Error occurred with &5 API. Reason code is &1.

API Introduced: V3R6

Top | Office APIs | APIs by category

Retrieve Office Programs (QOGRTVOE) API

Required Parameter Group:

1	Receiver variable	Output	Char(*)
2	Length of receiver variable	Input	Binary(4)
3	Format name	Input	Char(8)
4	Error code	I/O	Char(*)

Default Public Authority: *USE

Threadsafe: No

The Retrieve Office Programs (QOGRTVOE) API retrieves the programs that are used for office document requests and document conversion requests. The default value for the programs being retrieved is *IBM, which indicates that OfficeVision processing should be used for the request or conversion.

Authority

To use this program you need *SECADM authority.

Required Parameter Group

Receiver variable

OUTPUT; CHAR(*)

The variable to receive exit program information. This area must be large enough to accommodate the information specified.

Length of receiver variable

INPUT; BINARY(4)

The length of the receiver variable. The length must be at least 8 bytes. If the variable is not large enough to hold the requested information, the data is truncated.

Format name

INPUT; CHAR(8)

The format of the program to be retrieved. The valid format is OGOE0100 (see OGOE0100 Format).

Error code

I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter.

OGOE0100 Format

The following list shows the format of the program to be retrieved. For detailed descriptions of the fields, see Field Descriptions.

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Bytes returned
4	4	BINARY(4)	Bytes available
8	8	CHAR(10)	Document handling program name
18	12	CHAR(10)	Document handling program library name
28	1C	CHAR(1)	Document handling program supports mail flag
29	1D	CHAR(10)	Document conversion program name
39	27	CHAR(10)	Document conversion program library name
49	31	CHAR(1)	Application enabler support active flag
50	32	CHAR(1)	Mail handling support active flag
51	33	CHAR(1)	PCFILE identification active flag
52	34	CHAR(1)	Application enabler support activated flag
53	35	CHAR(1)	Mail handling support activated flag
54	36	CHAR(1)	PCFILE identification activated flag
55	37	CHAR(1)	Application enabler support activating flag
56	38	CHAR(1)	Mail handling support activating flag
57	39	CHAR(1)	PCFILE identification activating flag

Field Descriptions

Application enabler support active flag. An indication of whether the application enabler is active. The application enabler support is active if there is an application defined and the support has been made active.

0	The application enabler support will not be used with this system.
1	The application enabler support will be used with this system.

Application enabler support activated flag. An indication of whether the application enabler has been activated with the QOGCHGOE API.

0	The application enabler has not been activated.
1	The application enabler has been activated.

Application enabler support activating flag. An indication of whether the application enabler support can be activated. Application enabler support can become active if at least one application is defined and the support is made active with the QOGCHGOE API.

0	The application enabler would not become active if activated.
1	The application enabler would become active if activated.

Bytes available. The total length of all data available.

Bytes returned. The length of the data actually returned.

Document conversion program library name. The name of the library containing the program to be called for document conversions. This field will be blanks if the document conversion exit program name is *IBM.

Document conversion program name. The name of the program to be called for document conversions if an IBM conversion does not exist. If the program is *IBM, the IBM-supplied conversion exit or registered conversions will be called.

Document handling program library name. The name of the library containing the program to be called for document requests. This field will be blanks if the document handling exit program name is *IBM.

Document handling program name. The name of the program to be called for document requests. If the program is *IBM, OfficeVision is called for the document requests.

Document handling program supports mail flag. An indication of whether the document handling program supports mail requests (that is, MAILVIEW, MAILFWD, MAILREPLY, PRINT and PRINTOPTS from the Work with Mail display). If *IBM is specified as the document handling exit program, the value will be 1.

0	The exit program does not support mail.
1	The exit program does support mail.

Mail handling support active flag. An indication of whether the Mail Handling support is active. Mail handling support is active if there is an application defined that handles mail, application enabler support has been activated, and mail support has been activated.

0	The mail handling support is inactive with this application.
1	The mail handling support is active with this application.

Mail handling support activated flag. An indication of whether the mail handling support could be activated. It is activated with the QOGCHGOE API.

0	The mail handling support has not been activated.
1	The mail handling support has been activated.

Mail handling support activating flag. An indication of whether the mail handling support could be activated. Mail handling support is activated with the QOGCHGOE API.

0	The mail handling support cannot be activated.
1	The mail handling support can be activated.

PCFILE identification active flag. An indication of whether the PCFILE identification processing support is active.(2) If PCFILE identification support has indicated that at least one type has been defined and PCFILE identification support is active, then the PCFILE identification processing is active.

0	The PCFILE identification processing support is inactive with this application.
1	The PCFILE identification processing support is active with this application.

PCFILE identification activated flag. An indication of whether the PCFILE identification processing support could be activated. PCFILE identification is activated with the Change Office Programs (QOGCHGOE) API.

0	The PCFILE identification support has not been activated.
1	The PCFILE identification support has been activated.

PCFILE identification activating flag. An indication of whether the PCFILE identification processing could be activated. PCFILE identification support is activated with the QOGCHGOE API.

0	The PCFILE identification processing support would not become active if activated.
1	The PCFILE identification processing support would become active if activated.

Error Messages

Message ID	Error Message Text
CPF3C21 E	Format name &1 is not valid.
CPF3C24 E	Length of the receiver variable is not valid.
CPF3C90 E	Literal value cannot be changed.
CPF90A8 E	*SECADM special authority required to do requested operation.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.
OFC8019 E	Required module not on system.

API Introduced: V2R2

Search System Directory (QOKSCHD) API

Required Parameter Group:

1	Receiver variable	Output	Char(*)
2	Length of receiver variable	Input	Binary(4)
3	Format name of receiver variable	Input	Char(8)
4	Function	Input	Char(10)
5	Keep temporary resource indicator	Input	Char(1)
6	Request variable	Input	Char(*)
7	Length of request variable	Input	Binary(4)
8	Format name of request variable	Input	Char(8)
9	Error code	I/O	Char(*)

Default Public Authority: *USE

Threadsafe: No

The Search System Directory (QOKSCHD) API searches the system distribution directory and returns user information that matches the criteria input.

Authorities and Locks

To search the system directory, *USE authority is required to the QOKSCHD API.

Required Parameter Group

Receiver variable

OUTPUT; CHAR(*)

The receiver variable that is to receive the information requested. You can specify the size of the area smaller than the format requested as long as you specify the length of the receiver variable correctly. As a result, the API returns only the data the area can hold.

Length of receiver variable

INPUT; BINARY(4)

The length of the receiver variable.

Format name of the receiver variable

INPUT; CHAR(8)

The format of the information returned. The possible format names are:

SRCV0100	System directory user information. See SRCV0100 Format for more information on the SRCV0100 format.
SRCV0200	System directory field names. See SRCV0200 Format for more information on the SRCV0200 format.

Function

INPUT; CHAR(10)

The function name of this request. The functions supported are:

**SEARCH*	Search the system directory
**CLEANUP*	Clean up resources (if temporary resources were kept on previous search requests). Note that for cleanup, the keep temporary resource indicator parameter has to be 0.

Keep temporary resource indicator

INPUT; CHAR(1)

An indicator of whether to keep or delete the temporary resource that the search uses for search requests. This temporary resource keeps files open and other relevant information between calls. Performance will be better if this is done and you do multiple searches. If this API is called and this indicator is set to keep temporary resources, the last call should have the Function parameter set to *CLEANUP to close files and delete the resources.

0	Do not keep the temporary resource (and close the files).
1	Keep the temporary resource (and the files open).

The resource handle stores the information about the open files. When the search API is called again and the files have been left open, the resource handle needs to be passed. The resource handle differs from the continuation handle in that the resource handle saves information about open files. The same resource handle can be used to request different kinds of searches. If you are doing multiple types of searches, you do not want to do Function of *CLEANUP between the calls to the QOKSCHD API. Pass the same resource handle for these calls and it will make the searches perform faster. The continuation handle is used to continue the search with the same criteria because there are more users that match the search criteria. The continuation handle is documented in Field Descriptions.

Request variable

INPUT; CHAR(*)

The request variable that is the input buffer that requests the type of search to do.

Length of request variable

INPUT; BINARY(4)

The length of the request variable.

Format name of request variable

INPUT; CHAR(8)

The information requested. The possible format names are:

SREQ0100	System directory user information. See SREQ0100 Format for more information on the SREQ0100 format.

SREQ0200	System directory field names. See SREQ0200 Format for more information on the SREQ0200 format.

Error code

I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter.

SREQ0100 Format

The following table is the layout of the request variable, format SREQ0100, for requesting searches on the system directory user information.

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	CCSID of data input
4	4	BINARY(4)	Character set of data input
8	8	BINARY(4)	Code page of data input
12	C	CHAR(4)	Wildcard character
16	10	CHAR(1)	Convert receiver data indicator
17	11	CHAR(1)	Data to search
18	12	CHAR(1)	Run verify indicator
19	13	CHAR(1)	Continuation handle (input)
20	14	CHAR(16)	Resource handle
36	24	CHAR(8)	Format name of the search request array
44	2C	BINARY(4)	Offset to the search request array
48	30	BINARY(4)	Number of elements in the search request array
52	34	CHAR(8)	Format name of array of fields to return
60	3C	BINARY(4)	Offset to array of fields to return
64	40	BINARY(4)	Number of elements in the fields to return array.
68	44	CHAR(8)	Format name of array of users to return in the receiver variable
76	4C	BINARY(4)	Number of elements to return in the array of users to return
80	50	CHAR(8)	Format name of array of fields for each user returned
88	58	CHAR(8)	Format name of the order of field names to return
96	60	CHAR(1)	Return fields in order specified option
97	61	CHAR(3)	Reserved
		CHAR(*)	Array of fields to return
		CHAR(*)	Search request array

SREQ0101 Format

The following table is the search request array, format SREQ0101. This format name is used as input in the request variable, SREQ0100, in the field, format name of the search request array.

This is an array of search fields. There is a maximum of 100 entries.

The first element that is entered in the search request array is considered to be the key for the search. This is important for performance because the more distinct the key, the faster the search. Records will be returned in the order of the field specified as the key.

A performance consideration that should be made when choosing the key value for the value to match field is whether the field will contain a wildcard character. The search key should contain fields that occur a minimum number of times on the system. For this reason, it is more efficient to specify a key search value that does not contain wildcard characters. A wildcard character at the end of a search key value does not affect performance.

The search is an **AND** of all of the input fields. It will be a complete match or a generic match if a wildcard character is used as the last character of the field value.

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	BINARY(4)	Length of entry (including this field)
4	4	CHAR(1)	Compare value
5	5	CHAR(10)	Field name
15	F	CHAR(7)	Product ID
22	16	CHAR(1)	Case of data input
23	17	CHAR(1)	Reserved
24	18	BINARY(4)	Length of value
28	1C	CHAR(*)	Value to match

SREQ0102 Format

The following table is the array of fields to return, format SREQ102. This format name is used as input in the request variable, SREQ0100, in the field, format name of array of fields to return.

The SREQ0102 format provides for selecting a user controlled set of system distribution directory fields.

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	CHAR(10)	Field name
10	A	CHAR(7)	Product ID

SREQ0103 Format

The following table is the array of special values representing the fields to be returned, format SREQ0103. This format name is used in the request variable, SREQ0100, in the field, format name of the array of fields to return.

The SREQ0103 format provides for selecting a predefined set of system distribution directory fields. The allowed special values are *SYSDIR, *ORNAME, and *SMTP. See Field Descriptions for a definition of the special value of fields to be returned field.

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	CHAR(10)	Special value of fields to be returned

SREQ0200 Format

The following table is the layout of the request variable, format SREQ0200. This request returns the fields defined in the system distribution directory.

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	CHAR(1)	Type of system distribution directory fields to return
1	1	CHAR(17)	Continuation handle for directory fields (input)

SRCV0100 Format

The following table is the layout of the receiver variable, format SRCV0100. The receiver variable is the output buffer for the search results. This format defines the structure of the receiver variable.

Fields returned in the receiver variable will be in the order as documented in System directory field names. All of the fields with a product ID of *IBM will precede any fields with a product ID other than *IBM. Also, fields with a product ID of *IBM that appear in the array of fields to return more than once will be returned one time.

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	BINARY(4)	Number of bytes returned
4	4	BINARY(4)	Offset to order of fields returned array
8	8	BINARY(4)	Offset to the first array of users entry
12	C	BINARY(4)	Number of directory entries returned
16	10	CHAR(1)	Continuation handle (output)
17	11	CHAR(16)	Resource handle
		CHAR(*)	Array of users that match the search criteria
		CHAR(*)	Order of fields returned array

SRCV0101 Format

The following table is the structure of the array of users returned, format SRCV0101. This format name is specified in the request variable structure, SREQ0100, in the field, format name of array of users to return.

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	BINARY(4)	Length of data for this user
4	4	BINARY(4)	Number of fields returned
8	8	CHAR(*)	Array of fields for each user that was requested in the search request variable

SRCV0111 Format

The following table is the structure of the array of fields for each user returned, format SRCV0111. The format name is specified in the request variable structure, SREQ0100, in the field, format name of array of fields for each user returned. All fields requested are returned even if the fields are blank. The fields are returned in the order documented in System directory field names.

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	CHAR(10)	Field name
10	A	CHAR(7)	Product ID
17	11	CHAR(3)	Reserved
20	14	BINARY(4)	Character set or CCSID of the value
24	18	BINARY(4)	Code page of the value
28	1C	BINARY(4)	Length of field value returned
32	20	CHAR(*)	Field value

SRCV0112 Format

The following table is the structure of the array of fields for each user returned, format SRCV0112. The format name is specified in the request variable structure, SREQ0100, in the field, format name of array of fields for each user returned.

If you do not need user-defined fields or the field name and product ID, then request this format in SREQ0100 instead of SRCV0111, as it will save some overhead space. All fields requested are returned even if the fields are blank.

Format SRCV0120 tells the order in which the fields are returned. The order of the fields are always returned in the order as documented in System directory field names, but you may have a coding reason for needing to order the fields returned array.

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Character set or CCSID of the value
4	4	BINARY(4)	Code page of the value
8	8	BINARY(4)	Length of field value returned
12	C	CHAR(*)	Field value

SRCV0120 Format

The following table is the structure of the order of fields returned array, format SRCV0120. The format name is specified in the request structure, SREQ0100, in the field, format name of order of field names array returned.

Offset		Type	Field
Dec	Hex		
0	0	CHAR(10)	Field name
10	A	CHAR(7)	Product ID

SRCV0200 Format

The following table is the layout of the receiver variable, format SRCV0200. The receiver variable is the output buffer for the results from the system directory field name search request. The format name defines the structure of the receiver variable.

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Number of directory field names returned
4	4	BINARY(4)	Number of directory fields
8	8	BINARY(4)	Offset to the array of directory field names
12	C	CHAR(17)	Continuation handle for directory fields (output)
		CHAR(*)	Array of directory field names in the system distribution directory

Array of Directory Field Names

The following table is the structure of the field name array. This format contains the field names of the type that was requested in SREQ0200.

Offset		Type	Field
Dec	Hex		

0	0	CHAR(10)	Directory field name
10	A	CHAR(7)	Product ID
17	11	CHAR(10)	Directory field type
27	1B	CHAR(1)	Reserved
28	1C	BINARY(4)	Maximum field length

Field Descriptions

Array of directory field names in the system distribution directory. The array of fields that are returned as a result of the search request array. This array contains the directory field name, product ID, directory field type, and the maximum field length.

Array of fields for each user that was requested. The actual data for each field for the user returned as a result of the match. See either SRCV0111 Format, or SRCV0112 Format for the structure of this array.

Array of fields to return. The field names that the user wants returned in the receiver variable for each user found that matches the search criteria. See either SREQ0102 Format, or SREQ0103 Format for the structure of this array. There is no maximum number of elements.

Array of users that match the search criteria. The array of users that matches the search criteria on the request. See SRCV0101 Format for the structure of this array.

Case of data input. The case of the data input. The valid values are:

blank	The data of all fields except two is searched without regard to the case of the data (case-insensitive search). The exceptions are the IBM fields SMTPUSRID and SMTPRTE, which are searched with regard to the case of the data (case-sensitive search).
0	The same as *blank*, the preceding value description.
1	The data of all fields is searched without regard to the case of the data (case-insensitive search). **Note:** When you use the case of data input field for the SMTPUSRID or SMTPRTE field names, always qualify these field names with the product ID *IBM.

CCSID of data input. The CCSID of the data input or a special value. The special values are:

0	The default job CCSID will be used.
-1	The character set and code page will be used.
65535	The default job CCSID will be used.

Character set of data input. The character set of the data input. This field is used if the CCSID of data input is -1.

Character set or CCSID of the value. If the conversion of the data (convert receiver data indicator) was set to 0, this is the character set of the field value as it exists on the system. If the convert receiver indicator was set to 1, this is either a character set or CCSID depending on what was specified in the request variable for CCSID of data input. If the convert receiver data indicator was set to 2, this is the CCSID of the field value.

Code page of data input. The code page of the data input. This field is used if the CCSID of data input is -1.

Code page of the value. If the conversion of the data (convert receiver data indicator) was set to 0, this is the code page of the field value as it exists on the system. If the convert receiver indicator was set to 1, this is either a code page or CCSID depending on what was specified in the request variable for CCSID of data input. Otherwise, the code page is set to 0.

Compare value. The way to compare the value given with the field value in the system distribution directory entry. Only the value, 1 (equal), is currently supported.

Continuation handle (input). This allows you to continue to return data on a search if there was not enough space available to return all the entries that matched the search criteria on a previous search.

This value is 0 on the first search request. If you are continuing a search, this value needs to be set from the continuation handle (output) field.

The resource handle must be set if you are going to continue a search.

To use the continuation handle, you must keep the temporary resource space by setting the keep temporary resource indicator field to 1.

Continuation handle (output). An indicator for more directory entries available. The valid values are:

0	No
1	Yes

If this continuation handle is 0, either all the records that matched the search criteria were returned to this structure, or there were more records but the search files were not requested to be left open.

If the continuation handle is 1, there are more records available in the directory that match the search criteria, but there was no room available in the return structure. You may request more data by using this value in the request variable along with the resource handle.

Continuation handle for directory fields (input). Allows you to continue to return data on a search of directory fields if there was not enough space available to return all of the directory fields that matched the request.

This value is blank on the first request. If you are continuing a search, this value needs to be set from the continuation handle for directory fields (output) field.

Continuation handle for directory fields (output). Contains a value if there are more directory fields available for the search criteria requested. You may request more directory fields by using this value in the request variable field, continuation handle for directory fields (input).

Convert receiver data indicator. The option to convert the field values in the array of users returned for each user in the receiver variable.

The valid values are:

0	Do not convert receiver data. All data returned will be returned in the character set and code page as the data exists on the system. The character set and code page of the receiver data is in the array of fields for each user. See either SRCV0111 Format, or SRCV0112 Format for more information on the array of fields for each user.
1	Convert the receiver data. All data will be returned in the CCSID or character set and code page specified in the request variable.

2	Do not convert receiver data. All data will be returned as the data exists on the system, but the character set and code page will be converted to a CCSID tag. The CCSID of the receiver data is in the array of fields for each user. See either SRCV0111 Format, or SRCV0112 Format for more information on the array of fields for each user.

Data to search. The type of data to search on this system.

The valid values are:

0	Local data
1	All data on the system including shadowed data

Directory field name. Depending on the request in the SREQ0200 format, either an IBM-defined or a user-defined field. OS/400 system distribution directory field names have a product ID of *IBM. All other field names are for user-defined fields.

Directory field type. The following values can be returned for the field type and is meaningful only for user-defined fields:

**DATA*	User-defined field is for data.
**MSFSRVLVL*	User-defined field is for a mail service level.
**ADDRESS*	User-defined field is for an address.

See SRCV0200 Format for the field types for user-defined.

Field name. The field name can be an IBM-defined field name, or a user-defined field name. To determine the user-defined field names defined on the system, either use the Change System Directory Attributes (CHGSYSDIRA) command with F4, or retrieve the fields using the SREQ0200 format.

The following is a list of predefined field names that can be specified in the field name field of the array of search fields and the array of fields to return.

Notes:

1. There is a maximum of 512 bytes for user-defined fields.
2. There is a performance impact when a mixture of system directory field names, SMTP field names, and user-defined fields are specified in the search request array.

System directory field names

Note: This is the order in which the field names are returned when the return fields in order specified option is not 1 (0 is the default). The field is truncated so that ending blanks are not returned. This group of field names is the *SYSDIR group when the SREQ0103 format is used. See SREQ0103 Format for more information on the SREQ0103 format.

USER	User profile. The maximum length is 10.
INDUSR	Indirect user. The maximum length is 1. This only applies for local directory entries. The valid values are: *1* Yes; *0* No

PRTCOVER	Print cover page. The maximum length is 1. This only applies for local directory entries. The valid values are: *1* Yes *0* No **Note:** The PRTCOVER field can only be returned. It cannot be searched on.
NFYMAIL	Mail notification. The maximum length is 3. The first byte value represents: *1* Specific types of mail *2* All mail *3* No mail For specific types of mail: ● The second byte represents priority, private, and important mail. The valid values are: *1* Yes *0* No ● The third byte represents messages. The valid values are: *1* Yes *0* No **Note:** The NFYMAIL field can only be returned. It cannot be searched on.
USRID	User ID (DEN). The maximum length is 8.
LCLDTA	Local data indicator. The maximum length is 1. The valid values are: *0* Local *1* Shadowed This indicates where this user was created (origination). If the user was created on this system, this indicator is 0. If the user was shadowed from another system, this indicator is 1. If a remote user was created on this system, this indicator is 0.
USRADDR	User address (DGN). The maximum length is 8.
SYSNAME	System name (REN). The maximum length is 8.
SYSGRP	System group (RGN). The maximum length is 8.
USRD	User description.

	Each description field has a maximum length of 50. If a user has only one description, it will be returned trimmed. If a user has more than one description, all of the descriptions will be returned in this field, with each description using a maximum of 50 characters. The length of the value will tell how long the field is and, in effect, how many descriptions exist. For example, if a user has 3 descriptions, the length of value field will be 150 (50 characters for each description).
	Note: When there is more than one description, blank truncation is not performed.
FSTNAM	First name. The maximum length is 20.
PREFNAM	Preferred name. The maximum length is 20.
MIDNAM	Middle name. The maximum length is 20.
LSTNAM	Last name. The maximum length is 40.
FSTPREFNAM	First or preferred name. The maximum length is 20.
	Note: This is an input-only field. It is not output.
FULNAM	Full name. The maximum length is 50.
TITLE	Job title. The maximum length is 40.
CMPNY	Company. The maximum length of is 50.
DEPT	Department. The maximum length is 10.
NETUSRID	Network user ID. The maximum length is 47.
TELNBR1	Telephone number 1. The maximum length is 26.
TELNBR2	Telephone number 2. The maximum length is 26.
FAXTELNBR	Fax telephone number. The maximum length is 26.
LOC	Location. The maximum length is 40.
BLDG	Building. The maximum length is 20.
OFC	Office. The maximum length is 16.
ADDR1	Mailing address line 1. The maximum length is 40.
ADDR2	Mailing address line 2. The maximum length is 40.
ADDR3	Mailing address line 3. The maximum length is 40.
ADDR4	Mailing address line 4. The maximum length is 40.
CCMAILADR	cc:Mail** address. The maximum length is 255.
CCMAILCMT	cc:Mail comment. The maximum length is 126.
TEXT	Text. The maximum length is 50.

MSFSRVLVL	Mail server framework service level.

The field for the mail service level. When requesting this field to be searched on, supply only the field name and product ID with the field name in the first 10 bytes and the product ID in the last 7 bytes. When this field is returned in SRCV0111 or SRCV0112, the mail service level value contains the following structure:

- CHAR(10)-Field name

  The field name can be a special value or a user-defined field name. The special values are:

**USRIDX*	User index
**SYSMS*	System message store
**DOMINO*	Lotus Domino mail database

- CHAR(7)-Product ID

  The product ID of a user-defined field. *NONE indicates that there is no product ID for the user-defined field.

- BINARY(4)-Length of mail service level

  If the value is 0, then the field did not exist in the directory, the field was blank, or the field name is a special value of *USRIDX, *SYSMS, or *DOMINO.

- CHAR(*)-The value of the mail service level field and product ID. This value will be blank for *USRIDX, *SYSMS, and *DOMINO. For user-defined fields, it is the value of that field in the system distribution directory.

  The maximum size is 512.

See either SRCV0111 Format or SRCV0112 Format for more information on either the SRCV0111 format or the SRCV0112 format.

PREFADR	Preferred address.

The field for the preferred address. When requesting this field to be searched on, supply only the field name and product ID, with the field name in the first 10 bytes and the product ID in the last 7 bytes. When this field is returned in SRCV0111 or SRCV0112, the preferred address value contains the following structure:

- CHAR(10)-Field name

  The field name can be a special value, a user-defined field name, or an IBM-defined field name. The special values are:

**USRID*	User ID/address
**ORNAME*	X.400 O/R name
**SMTP*	SMTP name

- CHAR(7)-Product ID

  The product ID of the field. *IBM indicates an IBM-defined field name. *NONE indicates that there is no product ID for the user-defined field.

- CHAR(4)-Address type value

- CHAR(8)-Address type name
- BINARY(4)-Length of preferred address value

  If the value is 0, either the field did not exist in the directory or the field was blank.
- CHAR(*)-The value of the preferred address field and the product ID.

  The following is the structure of this field based on the field name value:

**USRID*	User ID (DEN) in the first 8 bytes, user address (DGN) in the second 8 bytes, system name (REN) in the third 8 bytes, and system group (RGN) in the last 8 bytes.
**ORNAME*	X.400 O/R name is formatted from the X.400 O/R name fields that exist for this user. See the IBM-defined field ORNAME for the structure of this field. The maximum size for this field is 909 bytes.
**SMTP*	The maximum size of the SMTP user ID is 24 bytes. The maximum size changes to 64 bytes if SMTP names are converted using the Convert Name SMTP (CVTNAMSMTP) command. You can check if the CVTNAMSMTP command has been run with the WRKNAMSMTP command. You get error TCP9610 if CVTNAMSTMP has been run. The first part of the *SMTP structure does not change. The SMTP user ID is the first 24 bytes followed by either the SMTP domain or route for 256 bytes. To support the new SMTP user ID length, 64 bytes follow the SMTP domain or route portion. Both the old SMTP user ID and the new SMTP user ID contain the SMTP user ID except when the SMTP user ID has more than 24 bytes. In this case the old SMTP user ID is blank.
IBM-defined or user-defined field	The contents of the field will be the value of the field in the system distribution directory. The maximum size is 512 bytes.

See either SRCV0111 Format or SRCV0112 Format for more information on either the SRCV0111 format or the SRCV0112 format.

ALWSYNC Allow synchronization. The maximum length is 1.

This field indicates whether the user's entry should be synchronized with directories other than the System Distribution Directory. The valid values are:

1	Yes
0	No

DLOOWN	DLO owner. The maximum length is 10.

This field indicates if the user profile or the group profile will be assigned the ownership of the document library objects (DLOs) for this directory entry. The valid values are:

**USRPRF*	User profile
**GRPPRF*	Group profile

MGRCODE	Manager code. The maximum length is 1.

This field indicates whether the user is a manager. The valid values are:

1	Yes
0	No

X.400 O/R field names:

These fields are the group of fields for the *ORNAME special value in the SREQ0103 format.

ORNAME The paper representation of the X.400 O/R name. This field cannot be searched on, but it can be returned in the receiver variable.

The string starts with X.400 and is followed by the following formats:

- Non-DDA (domain-defined attributes) are represented as:

```
<attribute type>=<attribute value>
```

- DDA (domain-defined attributes) are represented as:

```
DDA.<DDA type>=<DDA value>
```

Abbreviations for the attributes in an X.400 O/R name are:

C	Country or region name
A	Administration Domain (ADMD) name
P	Private Management Domain (PRMD) name
O	Organization name
OU	Organizational unit name
OU1	Organizational unit 1 name
OU2	Organizational unit 2 name
OU3	Organizational unit 3 name
OU4	Organizational unit 4 name
S	Surname

G	Given name
I	Initials
GQ	Generation qualifier
DDA	Domain-defined attribute

When an O/R name contains only one organization unit attribute, the type OU is used. When an address contains more than one organization unit attribute, the types OU1, OU2, OU3, and OU4 are used. OU1 is the most significant designator in the organization's hierarchy.

Every attribute, except the last one is followed by a semicolon.

An example paper representation of an X.400 O/R name is:

```
X.400 C=US;A=ANYMAIL;P=XYZ;O=CLEANING COMPANY;
OU=SALES DEPT;S=DOE;G=John;I=JA;DDA.ID=123999
```

If each X.400 field is returned separately, then:

- COUNTRY would be US
- ADMD would be ANYMAIL
- PRMD would be XYZ
- ORG would be CLEANING COMPANY
- One of the organizational units would be SALES DEPT
- SURNAM would be DOE
- GIVENNAM would be JOHN
- INITIALS would be JA
- One of the domain-defined attribute types fields would be ID, and the value of that field would be 123999.

COUNTRY	Country or region. The maximum length is 3.
ADMD	Administration domain. The maximum length is 16.
PRMD	Private management domain. The maximum length is 16.
ORG	Organization. The maximum length is 64.
SURNAM	Surname. The maximum length is 40.
GIVENNAM	Given name. The maximum length is 16.
INITIALS	Initials. The maximum length is 5.
GENQUAL	Generation qualifier. The maximum length is 3.
ORGUNIT1	Organization unit 1. The maximum length is 32.
ORGUNIT2	Organization unit 2. The maximum length is 32.
ORGUNIT3	Organization unit 3. The maximum length is 32.
ORGUNIT4	Organization unit 4. The maximum length is 32.
DMNDFNAT1	Domain-defined attribute type 1. The maximum length is 8.
DMNDFNAV1	Domain-defined attribute value 1. The maximum length is 128.
DMNDFNAT2	Domain-defined attribute type 2. The maximum length is 8

DMNDFNAV2	Domain-defined attribute value 2. The maximum length is 128.
DMNDFNAT3	Domain-defined attribute type 3. The maximum length is 8.
DMNDFNAV3	Domain-defined attribute value 3. The maximum length is 128.
DMNDFNAT4	Domain-defined attribute type 4. The maximum length is 8.
DMNDFNAV4	Domain-defined attribute value 4. The maximum length is 128.

See SREQ0103 Format for more information on the SREQ0103 format.

Simple Mail Transfer Protocol (SMTP) field names:

These fields are the group of fields for the *SMTP special value in the SREQ0103 format.

SMTPUSRID	SMTP user ID. The maximum length is 24. However, the maximum length changes to 64 if the SMTP names are converted using the CVTNAMSMTP command. **Note:** This field is a case-sensitive field.
SMTPDMN	SMTP domain. Either the SMTP domain or the SMTP route can exist, not both. The maximum length is 256.
SMTPRTE	SMTP route. Either the SMTP domain or the SMTP route can exist, not both. The maximum length is 256.

See SREQ0103 Format for more information on the SREQ0103 format.

Field value. The value of the field.

Format name of array of fields for each user returned. The format name of the array of fields for each user returned. This is the specific structure for each user in either the SRCV0111 format or the SRCV0112 format. See either SRCV0111 Format, or SRCV0112 Format for more information on the array of fields for each user.

Format name of array of fields to return. The format name of the array of fields to return so that the fields of each user found matching the search criteria that is needed is known. The possible values are SREQ0102 and SREQ0103. See SREQ0102 Format, or SREQ0103 Format for more information on the array of fields to return for each user.

Format name of array of users to return in the receiver variable. The format name of the array of users to return after the search is complete. This is so the format to return the search results is known. The possible value is SRCV0101. See SRCV0101 Format for more information on the array of users returned.

Format name of order of field names array returned. The format name of the order of field names to be returned. Possible values are blank and SRCV0120. If the value is blank, the order of the field names array is not returned. The order of field names array is not returned if user-defined fields are requested as only the user-defined fields that exist for each user is returned and are not in any specified order.

Format name of the search request array. The format name of the search request array so that the format of the search request is known. The possible value is SREQ0101. See SREQ0101 Format for more information on the array of search fields.

Length of data for this user. The length of both the header information and the fields that are returned for the user.

Length of entry (including this field). The length of the entry in the array of all of the fields including this field.

Length of field value returned. The length, in bytes, of the field value. If this value is 0, there is no data in the field.

Length of value. If the length of the value is 0, this field is ignored on the search request. There must be at least one field in the search request array with a length of value greater than 0 and the value is not blank spaces.

The maximum length is 512.

Maximum field length. The maximum length of this directory field.

Number of bytes returned. The length of the data returned. If the receiver variable is not large enough to hold all of the data available, only the data that will fit in the space available is returned and this value will be less than the bytes available.

Number of directory entries returned. This is the number of directory entries that are returned in SRCV0101. It is the number of directory entries found as a result of the search request. If more directory entries are available but there was not enough space available in the SRCV0101 structure, the continuation handle will be 1. If all of the entries are returned, the continuation handle will be 0.

Number of directory field names returned. The number of elements in the field name array.

Number of directory fields. The total number of directory fields for the type of fields requested. If the continuation handler for directory fields is not blank, then there are more fields available.

Number of elements in the fields to return array. The number of fields to be returned for each user. These fields are returned in the receiver variable.

Number of elements in the search request array. The number of fields and values to match for the search request. The maximum number of elements allowed is 100.

Number of elements to return in the array of users to return. The number that should be used to return a fixed number of elements on a call to the API. The number of elements returned will be less than or equal to the number specified here. If the number returned is less than the number specified, then either all of the elements that met the criteria were returned or the size of the receiver variable could not hold the number specified.

The continuation handle will be set if more data exists after returning the number of elements specified for this parameter.

If this number is 0, the maximum number of elements possible is returned.

This field could be used to limit the run time of the API. For instance, it could be used to retrieve only the number of users needed to file a subfile page.

Number of fields returned. The number of fields returned for the user. Fields are returned even if blank.

Offset to array of fields to return. The offset, in bytes, to the array of fields to return.

Offset to order of fields returned array. The offset, in bytes, to the order of fields returned array. The value can be 0 if either the order of the fields is not needed or the user-defined fields are returned. This is requested in the SREQ0100 format. See SREQ0100 Format for more information on the offset to order of fields returned array.

Offset to the array of directory field names. The offset, in bytes, to the array of directory field names at the bottom of the SRCV0200 structure.

Offset to the first array of users entry. The offset, in bytes, to the array of users in the SRCV0100 structure. This is the offset to the first entry in the array of users that are returned that match the search criteria.

Offset to the search request array. The offset, in bytes, to the search request array in the SREQ0100 format. See SREQ0100 Format for more information on the offset to the search request array.

Order of fields returned array. The array of fields names of the order of the fields in the array of fields for each user (in either the SRCV0111 format or the SRCV0112 format) that is returned. For more information see SRCV0120 Format.

Product ID. A special value of *IBM indicates that the field is an OS/400 system distribution directory field. A value of blank spaces indicates that there is no product ID associated with the field name.

An error will be signaled if:

- The field name and product ID cannot be found.
- The product ID is blank and the field name does not exist with a product ID of *NONE.

Reserved. This is used to align the character set of the value field.

Resource handle. This value needs to be set from the output resource handle variable that was used as output in the receiver buffer from a previous search.

On the first call, this value needs to be blank.

To use the resource handle, you must specify to keep the temporary resources option on the previous search.

If you are continuing a search, the resource handle must be set from the output resource handle variable from a previous search of the same criteria.

If you are not continuing a search, the resource handle can be reused and the continuation handle will be set to 0. This will reuse the open files and temporary work areas.

If a resource handle is reused on another search, a continuation of a previous search is attempted using the same resource handle and the continuation is no longer valid for the previous search.

Return fields in order specified option. The option to return the fields in the order specified in the array of fields to return format. See SREQ0102 Format for the structure of this array. The valid values are:

0	Return fields in the predefined order as specified in the Field name field description in Field name. This is the default.
1	Return the fields as specified in SREQ0102 Format. This option may give slower response on the search request. SRCV0120 Format and SREQ0103 Format cannot be used with this option.

Run verify indicator. The option to verify the input parameters. Running the verification process has the potential to increase the time it takes to run a search. If the verification flag is set off and no valid data is passed as input, unpredictable results may occur.

The purpose of this indicator is so verification does not have to be run when continuing the return of data, or when a search request is repeated.

The valid values are:

0	Do not run the verification process.
1	Run the verification process.

For the first search request, it is recommended that this value be 1.

Search request array. An array of search fields. There is a maximum of 100 entries. See SREQ0101 Format for the structure of this array.

The first element that is entered in the search request array is considered to be the key for the search. This is important for performance because the more distinct the key, the faster the search. Records will be returned in the order of the field specified as the key.

A performance consideration that should be made when choosing the key value is a value to match that contains wildcard characters. The search key should contain fields that occur a minimum number of times on the system. For this reason, it is more efficient to specify a key search value that does not contain wildcards.

The search is an AND of all the input fields. It will be a full match or a generic match if a wildcard character is used as the last character of the field value.

Special value of fields to be returned. This field selects a group of fields depending on the special value. The order of the field names returned in the SRCV0101 format is returned in the format, SRCV0120. The SRCV0120 format is optional. It is optional because the order of the field names is always the same, but it is given so you can programmatically find the order of fields returned.

The special values are:

**SYSDIR*	Returns all of the system distribution directory fields except the X.400 O/R name fields, the SMTP name fields, and the user-defined fields.
**ORNAME*	Returns the X.400 O/R name fields.
**SMTP*	Returns the SMTP name fields.

Type of system distribution directory fields to return. The indicator for the type of system distribution directory fields that are defined on the system to be returned.

The possible values are:

0	All system distribution directory fields, including both IBM-defined fields and user-defined fields.
1	IBM-defined system distribution directory fields.
2	User-defined system distribution directory fields.

Value to match. If a blank value is encountered, the API will ignore the field.

All values entered are monocased and converted to a common character set and code page before the search takes place. Therefore, the uppercase and lowercase characters are not distinguished, except for the SMTP user ID where it is distinguished. However, if the SMTP names are converted (CVTNAMSMTP command), the SMTP user ID is not distinguished because the SMTP fields are converted into user-defined fields.

A maximum of one wildcard character is allowed in the value to match field.

Wildcard character. The wildcard character indicates the wildcard character used for wildcard searches. An example of a wildcard character is an asterisk (*); however, the wildcard character can be anything you specify. If the character is encountered in the values to search on, that value indicates a wildcard search. This wildcard character will be in the CCSID or character set and code page (CHRID) that is used as input.

The wildcard character represents any number of characters (not just one character) in the search value to match. If a blank is used for a wildcard value, the API will not check for wildcards and the value will be checked specifically as input.

Note: The wildcard character field is a CHAR(4) because of DBCS and SBCS. In addition, for SBCS the wildcard character field is left-adjusted.

Error Messages

Message ID	Error Message Text
CPF3C90 E	Literal value cannot be changed.
CPF3CF1 E	Error code parameter not valid.
CPF9A8C E	Error occurred with QOKSCHD API. Reason code &1.
CPF9845 E	Error occurred while opening file &1.
CPF9846 E	Error while processing file &1 in library &2.
CPF9847 E	Error occurred while closing file &1 in library &2.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.
CPI9A9C I	Search data does not exist.

API Introduced: V3R1

Directory Maintenance Exit Program

Required Parameter Group:

1	Request type	Input	Char(10)
2	Directory information format	Input	Char(10)
3	Owning system name	Input	Char(8)
4	User making request	Input	Char(10)
5	System making request	Input	Char(8)
6	Length of directory information	Input	Binary(4)
7	Directory information	Input	Char(*)
8	User exit program type	Input	Char(10)
9	Field name and product ID in error	Output	Char(17)

```
QSYSINC Member Name:  EOKDRVF

Exit Point Name:  QIBM_QOK_VERIFY
                  QIBM_QOK_NOTIFY

Exit Point Format Name:  VRFY0100
```

The Directory Maintenance exit program allows the administrator to make decisions based on directory entry additions, changes, or deletions. Two exit points are available.

The maintenance program registered at exit point QIBM_QOK_VERIFY is called before any directory entry, department, or location is added, changed, or removed from the system. This exit program is more specifically known as the verification maintenance exit program. The verification maintenance exit program allows the administrator to define additional security or syntax checking on the data.

The maintenance program registered at exit point QIBM_QOK_NOTIFY is called after any directory entry, department, or location is added, changed, or removed from the system. This exit program is more specifically known as the notification maintenance exit program.

The verification maintenance exit program is specified on the VRFPGM parameter of the Change System Directory Attribute (CHGSYSDIRA) command. The verification maintenance exit program can also be specified using the Work with Registration Information (WRKREGINF) command. The notification maintenance exit program can only be specified using the Work with Registration Information (WRKREGINF) command.

Exit programs that have been registered through the registration facility for common exit programs can be viewed by using the Work with Registration Information (WRKREGINF) command.

The maintenance programs are given all updated information known about the directory entry, department, or location. The verification maintenance exit program returns to the directory service an indication as to whether the add, change, or delete operation is to be applied to the Enterprise Address Book (EAB). On the iSeries server, this is called the system distribution directory. The EAB is a collection of data, such as information about people, departments, and locations in a network. An example of an enterprise is a company.

Whether or not update requests that are rejected by the verification maintenance exit program are supplied to other systems depends on the origin of the update.

- When a local update request is rejected by the verification maintenance exit program, the update does not affect the local system. If the iSeries server is participating in directory shadowing, the update is not supplied to other systems in the directory shadowing network.

- If the iSeries server is participating in directory shadowing and a shadowed update request is received and is rejected by the verification maintenance exit program, then the update does not affect the local system.

  Other systems in the directory shadowing network may be affected. The results depend on the nature of the shadowed update request:

  - When an add request is rejected, it is filtered out of (does not appear on) the local system, but remains on the network. Information about the add request is stored separately in a change log and will subsequently be supplied to other systems in the directory shadowing network.
  - When a change or delete request is rejected, the local system keeps its original information. In addition, the original information will be supplied again to other systems, including the system the information originated from, to keep directory information consistent throughout the network.

The verification criteria should be consistent on all your systems to help reduce the amount of processing on the network.

The system can get back the data which was filtered, but it is not a simple task. In order to retrieve the data which has been filtered out, the data can be shadowed again from the system where the data resides.

Required Parameter Group

Function being requested

INPUT; CHAR(10)

The type of operation that the user is requesting to do to the directory information that is described by the other parameters.

**ADD*	Information is being added.
**ADDDSC*	User description is being added.
**CHG*	Information is being changed. All fields in the directory information that are not changed will be X'00' except for the first field of every table. That field indicates what directory entry, department, or location is being changed.
**DLT*	Information is being deleted.
**DLTDSC*	User description is being deleted.

Directory information format

INPUT; CHAR(10)

The format of the directory information that is being worked with. The information is provided in the directory information parameter. The valid formats are:

CHKP0100	Directory entry (See CHKP0100 Format.)

CHKP0200	Department entry (See CHKP0200 Format.)
CHKP0300	Location entry (See CHKP0300 Format.)

These formats have the same layout as the SUPP0100, SUPP0200, and SUPP0300 formats used for the Directory Supplier exit program. This allows a single program to be used as both a verification maintenance program and a supplier program.

Owning system name

INPUT; CHAR(8)

The name of the system that "owns" the directory entry, department, or location that is being worked with. The owning system is the system that originally added the data to the network.

**LOCAL*	The entry is owned by the local system.

User making request

INPUT; CHAR(10)

The user profile name of the user that is doing the request. When using the shadowing function, this is the user that originated the modification.

System making request

INPUT; CHAR(8)

The system from which the request is coming. When using the shadowing function, this is the system that originated the modification.

Length of directory information

INPUT; BINARY(4)

The length of the directory information in the directory information parameter. The length depends on the directory information format. Each format has a different (but fixed) length as shown in specific format tables.

Directory information

INPUT; CHAR(*)

The directory information that is associated with the directory entry, department, or location that the request is made against. For the format of this character parameter, refer to the specific format table (CHKP0100 Format , CHKP0200 Format , or CHKP0300 Format) and to the Field Descriptions.

User exit program type

INPUT; CHAR(10)

The user exit program type that is associated with the call of the Directory Maintenance exit program. This parameter is provided so that a single program can be used as both a maintenance program and a supplier program. This parameter is set to *VRFPGM if calling the verification maintenance exit program or *NFYPGM if calling the notification maintenance exit program.

Field name and product ID in error

OUTPUT; CHAR(17)

The field name and product ID that caused the CPF89A4 error. The first 10 characters are the field name; the second 7 characters contain the product name. The product ID can have the following special values:

**NONE*	A field name that does not have a product ID.
**IBM*	A field defined by the OS/400 system distribution directory.

This parameter is used only by the verification maintenance exit program. The notification maintenance exit program does not use any output parameter. This is because the information has already been added to the directory and the notification maintenance exit program cannot reject it.

This field will be recognized only for an add or a change of a directory entry where the Work with Directory Entries (WRKDIRE) panel support is being used. The field will be highlighted and the cursor will be positioned on the field. The field will be ignored under any other conditions.

The field name can be a user-defined name or a name supplied by the OS/400 system distribution directory. To display the user-defined names, either type CHGSYSDIRA and press F4, or use the SREQ0200 format of the QOKSCHD API. For more information on the SREQ0200 format of the QOKSCHD API, see SREQ0200 Format.

The following names are defined by the OS/400 system distribution directory. The product ID of these names is *IBM.

LSTNAM	Last name
FSTNAM	First name
MIDNAM	Middle name
PREFNAM	Preferred name
FULNAM	Full name
DEPT	Department
USRID	User ID (DEN)
USRADDR	User address (DGN)
USRD	User description
SYSNAME	System name (REN)
SYSGRP	System group (RGN)
USER	User profile
NETUSRID	Network user ID
TELNBR1	Telephone number 1
TELNBR2	Telephone number 1
FAXTELNBR	Fax telephone number
LOC	Location
BLDG	Building
CMPNY	Company
OFC	Office
TITLE	Job title
ADDR1	Mailing address line 1
ADDR2	Mailing address line 2

ADDR3	Mailing address line 3
ADDR4	Mailing address line 4
INDUSR	Indirect user
LCLDTA	Local data indicator
TEXT	Text
COUNTRY	Country or region
ADMD	Administration domain
PRMD	Private management domain
ORG	Organization
SURNAM	Surname
GIVENNAM	Given name
INITIALS	Initials
GENQUAL	Generation qualifier
ORGUNIT1	Organization unit 1
ORGUNIT2	Organization unit 2
ORGUNIT3	Organization unit 3
ORGUNIT4	Organization unit 4
DMNDFNAT1	Domain-defined attribute type 1
DMNDFNAV1	Domain-defined attribute value 1
DMNDFNAT2	Domain-defined attribute type 2
DMNDFNAV2	Domain-defined attribute value 2
DMNDFNAT3	Domain-defined attribute type 3
DMNDFNAV3	Domain-defined attribute value 3
DMNDFNAT4	Domain-defined attribute type 4
DMNDFNAV4	Domain-defined attribute value 4
CCMAILADR	cc:Mail address
CCMAILCMT	cc:Mail comment
MSFSRVLVL	Mail server framework service level
PREFADR	Preferred address
ALWSYNC	Allow synchronization
DLOOWN	DLO owner

Rejecting an Update Operation

The user-written verification maintenance exit program may reject update requests. To do so, the verification maintenance exit program must signal a specific program message to the directory services module that called it, and then return. An update is rejected based on the restrictions that have been identified. For example, if three people have authority to make updates but the program allows only one user to do updates, then update requests by all other users are rejected.

To allow a directory update request, the verification maintenance exit program only returns to the program that called it. Two program messages have been defined for the purpose of rejecting directory updates. The messages are:

CPF89A3	Operation not successful due to authority reasons.
CPF89A4	Operation not successful due to data validation reasons.

Note: The message must be signalled as an escape message. A diagnostic or informational message can be signalled before the escape message to give additional information about the error.

You may provide an optional data structure with message variable substitution text. This will help clarify the reasons for the rejection to the users. The optional data structure is:

CHAR(10)	The profile name of the user who requested (entered) the update. This information is from the user making request parameter.
CHAR(8)	The system name of the user who requested (entered) the update. You can get this from the system making request parameter.
CHAR(120)	A description of the reason the exit program is rejecting the request.

CHKP0100 Format

Offset			
Dec	**Hex**	**Type**	**Field**
Note: The following fields are in code page 500 and character set 697.			
0	0	CHAR(16)	User ID/address
16	10	CHAR(16)	System name/group
32	20	CHAR(10)	User profile
42	2A	CHAR(47)	Network user ID
89	59	CHAR(16)	New user ID/address
105	69	CHAR(16)	Old user to forward from user ID/address
121	79	CHAR(1)	Indirect user
122	7A	CHAR(1)	Print personal mail
123	7B	CHAR(3)	Reserved
Note: The character sets and code pages immediately follow the individual fields in the list below.			
126	7E	CHAR(50)	Description
176	B0	BINARY(4)	Character set
180	B4	BINARY(4)	Code page
184	B8	CHAR(40)	Last name
224	E0	BINARY(4)	Character set
228	E4	BINARY(4)	Code page
232	E8	CHAR(20)	First name
252	FC	BINARY(4)	Character set
256	100	BINARY(4)	Code page

260	104	CHAR(20)	Middle name
280	118	BINARY(4)	Character set
284	11C	BINARY(4)	Code page
288	120	CHAR(20)	Preferred name
308	134	BINARY(4)	Character set
312	138	BINARY(4)	Code page
316	13C	CHAR(2)	Reserved
318	13E	CHAR(50)	Full name
368	170	BINARY(4)	Character set
372	174	BINARY(4)	Code page
376	178	CHAR(2)	Reserved
378	17A	CHAR(10)	Department
388	184	BINARY(4)	Character set
392	188	BINARY(4)	Code page
396	18C	CHAR(2)	Reserved
398	18E	CHAR(50)	Job title
448	1C0	BINARY(4)	Character set
452	1C4	BINARY(4)	Code page
456	1C8	CHAR(2)	Reserved
458	1CA	CHAR(50)	Company
508	1FC	BINARY(4)	Character set
512	200	BINARY(4)	Code page
516	204	CHAR(2)	Reserved
518	206	CHAR(26)	Telephone number 1
544	220	BINARY(4)	Character set
548	224	BINARY(4)	Code page
552	228	CHAR(2)	Reserved
554	22A	CHAR(26)	Telephone number 2
580	244	BINARY(4)	Character set
584	248	BINARY(4)	Code page
588	24C	CHAR(40)	Location
628	274	BINARY(4)	Character set
632	278	BINARY(4)	Code page
636	27C	CHAR(20)	Building
656	290	BINARY(4)	Character set
660	294	BINARY(4)	Code page
664	298	CHAR(16)	Office
680	2A8	BINARY(4)	Character set
684	2AC	BINARY(4)	Code page
688	2B0	CHAR(40)	Mailing address line 1
728	2D8	BINARY(4)	Character set
732	2DC	BINARY(4)	Code page

736	2E0	CHAR(40)	Mailing address line 2
776	308	BINARY(4)	Character set
780	30C	BINARY(4)	Code page
784	310	CHAR(40)	Mailing address line 3
824	338	BINARY(4)	Character set
828	33C	BINARY(4)	Code page
832	340	CHAR(40)	Mailing address line 4
872	368	BINARY(4)	Character set
876	36C	BINARY(4)	Code page
880	370	CHAR(2)	Reserved
882	372	CHAR(50)	Text
932	3A4	BINARY(4)	Character set
936	3A8	BINARY(4)	Code page
940	3AC	CHAR(1)	Print cover page
941	3AD	CHAR(1)	Mail notification
Note: The following X.400 fields are in the character set and code page as defined by 1984 X.400 standards.			
942	3AE	CHAR(3)	X.400 country or region
945	3B1	CHAR(16)	X.400 administration domain
961	3C1	CHAR(16)	X.400 private domain
977	3D1	CHAR(64)	X.400 organization
1041	411	CHAR(40)	X.400 surname
1081	439	CHAR(16)	X.400 given name
1097	449	CHAR(5)	X.400 initials
1102	44E	CHAR(3)	X.400 generation qualifier
1105	451	CHAR(32)	X.400 organization unit 1
1137	471	CHAR(32)	X.400 organization unit 2
1169	491	CHAR(32)	X.400 organization unit 3
1201	4B1	CHAR(32)	X.400 organization unit 4
1233	4D1	CHAR(8)	X.400 domain attribute type 1
1241	4D9	CHAR(128)	X.400 domain attribute value 1
1369	559	CHAR(8)	X.400 domain attribute type 2
1377	561	CHAR(128)	X.400 domain attribute value 2
1505	5E1	CHAR(8)	X.400 domain attribute type 3
1513	5E9	CHAR(128)	X.400 domain attribute value 3
1641	669	CHAR(8)	X.400 domain attribute type 4
1649	671	CHAR(128)	X.400 domain attribute value 4
1777	6F1	CHAR(3)	Reserved
1780	6F4	CHAR(32)	Fax telephone number
1812	714	BINARY(4)	Character set
1816	718	BINARY(4)	Code page
1820	71C	CHAR(17)	Mail service level

1837	72D	CHAR(29)	Preferred address
1866	74A	CHAR(255)	cc:Mail address
2121	849	CHAR(126)	cc:Mail comment
2247	8C7	CHAR(1)	Allow synchronization
2248	8C8	BINARY(4)	Offset to user-defined fields array
2252	8CC	BINARY(4)	Number of elements (fields) in user-defined fields array
2256	8D0	CHAR(10)	DLO owner
Note: All fields that are not changed will be X'00' except for the user ID/address field.			

Array for User-Defined Fields

The following table is the array for user-defined fields, format CHKP0100.

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	BINARY(4)	Displacement to next user-defined field element in this array
4	4	CHAR(10)	Field name
14	E	CHAR(7)	Product ID
21	15	CHAR(3)	Reserved
24	18	BINARY(4)	Character set
28	1C	BINARY(4)	Code page
32	20	BINARY(4)	Length of field value returned
36	24	CHAR(*)	Field value

CHKP0200 Format

Offset			
Dec	**Hex**	**Type**	**Field**
Note: The character sets and code pages immediately follow the individual fields in the list below.			
0	0	CHAR(2)	Reserved
2	2	CHAR(10)	Department
12	C	BINARY(4)	Character set
16	10	BINARY(4)	Code page
20	14	CHAR(2)	Reserved
22	16	CHAR(50)	Title
72	48	BINARY(4)	Character set
76	4C	BINARY(4)	Code page

80	50	CHAR(2)	Reserved
82	52	CHAR(10)	Reports to department
92	5C	BINARY(4)	Character set
96	60	BINARY(4)	Code page
Note: The following field is in code page 500 and character set 697.			
100	64	CHAR(16)	Manager user ID/address
116	74	CHAR(2)	Reserved
118	76	CHAR(10)	Old department
128	80	BINARY(4)	Character set
132	84	BINARY(4)	Code page
Note: All fields that are not changed will be X'00' except for the department field and its corresponding character set and code page fields.			

CHKP0300 Format

Offset			
Dec	**Hex**	**Type**	**Field**
Note: The character sets and code pages immediately follow the individual fields in the list below.			
0	0	CHAR(40)	Location
40	28	BINARY(4)	Character set
44	2C	BINARY(4)	Code page
48	30	CHAR(2)	Reserved
50	32	CHAR(30)	Location line 1
80	50	BINARY(4)	Character set
84	54	BINARY(4)	Code page
88	58	CHAR(2)	Reserved
90	5A	CHAR(30)	Location line 2
120	78	BINARY(4)	Character set
124	7C	BINARY(4)	Code page
128	80	CHAR(2)	Reserved
130	82	CHAR(30)	Location line 3
160	A0	BINARY(4)	Character set
164	A4	BINARY(4)	Code page
168	A8	CHAR(2)	Reserved
170	AA	CHAR(30)	Location line 4
200	C8	BINARY(4)	Character set
204	CC	BINARY(4)	Code page
208	D0	CHAR(2)	Reserved
210	D2	CHAR(30)	Location line 5

240	F0	BINARY(4)	Character set
244	F4	BINARY(4)	Code page
248	F8	CHAR(2)	Reserved
250	FA	CHAR(30)	Location line 6
280	118	BINARY(4)	Character set
284	11C	BINARY(4)	Code page
288	120	CHAR(40)	Location changed to
328	148	BINARY(4)	Character set
332	14C	BINARY(4)	Code page
336	150	CHAR(40)	Old location
376	178	BINARY(4)	Character set
380	17C	BINARY(4)	Code page
Note: All fields that are not changed will be X'00' except for the location field and its corresponding character set and code page fields.			

Field Descriptions

Allow synchronization. Whether the directory entries should be synchronized with directories other than the system distribution directory. The values are 0 for no and 1 for yes.

Building. The name or number that identifies the user's building.

cc:Mail address. The cc:Mail address for a user. This field has a maximum of 126 characters, or 255 if the cc:Mail address contains both a remote post office name and an alias name.

cc:Mail comment. The cc:Mail comment for this user.

Character set. The character identifier (graphic character set) that was used by the work station to enter the data for the field.

Code page. The value specified on this parameter is used to instruct the printer device to interpret the hexadecimal byte string to print the same characters that were intended when the text was created.

Company. The name of the company for whom the user works.

Department. The name or number that identifies the user's department.

Description. The description associated with the user ID. One entry in the directory can have several different descriptions.

Displacement to next user-defined field element in this array. The displacement, in bytes, to the next user-defined field. Use this value to increment the pointer to get to the next user-defined field.

DLO owner. A special value indicating whether the user profile or the group profile will be assigned ownership of newly created document library objects (DLOs) associated with this directory entry.

Fax telephone number. The facsimile telephone number.

Field name. The user-defined field name.

Field value. The value of the user-defined field. The maximum is 512 bytes.

First name. The user's first name or given name.

Full name. The user's full name as it appears when a directory is viewed or searched.

Indirect user. A user enrolled in the system distribution directory who receives mail but never signs on to view it. An indirect user receives printed mail only. The values are 0 for no and 1 for yes.

Job title. The title of the user's occupation.

Last name. The user's last name.

Length of field value. The length of the user-defined field value. If the value is 0, there is no data in that field for this user.

Location. The location of the business or system. Some examples of location are city, state, or street address.

Location changed to. The new location value after combining locations.

Location lines 1 through 6. The location of the business or system. These fields further describe a location name. For example, the field may contain the general mailing address for the location.

Mail notification. Whether or not the user wants to be notified when mail arrives.

The user can specify these values:

Blank	Notified for priority or personal mail and messages
1	Notified for priority or personal mail and messages
2	Notified for only priority or personal mail
3	Notified for messages only
4	Notified for all mail
0	No notification for mail

Mail service level. A 17-byte field where the first 10 bytes are the field name and the last 7 bytes are the product ID. The values for the mail service level can be:

**USRIDX*	User index
**SYSMS*	System message store
**DOMINO*	Lotus Domino mail database
A user-defined field name and product ID	The product ID will be blank if the product ID does not exist.

Mailing address lines 1 through 4. The address of the user.

Manager user ID/address. The department manager's user ID and address.

Middle name. The user's middle name.

Network user ID. A unique value associated with each user in the Enterprise Address Book. For example, the value could be the user ID/address, the social security number, or the employee number.

New user ID/address. The new user ID and address used during the rename operation.

Office. The name or number that identifies the user's office.

Old department. The previous department value before being changed.

Old location. The previous location value before being changed.

Old user to forward from user ID/address. This field shows the previous user ID and address from which the user forwards mail.

Preferred address. A 29-byte field where the first 10 bytes are the field name, the second 7 bytes are the product ID, the third 4 bytes are reserved, and the last 8 bytes are the address type name.

The special values of the field name in preferred address can be:

**USRID*	User ID/Address
**ORNAME*	X.400 O/R name
**SMTP*	SMTP name
A user-defined or IBM-defined field.	The product ID is *IBM for an IBM-defined field. The product ID will be blank if the product ID does not exist.

Preferred name. The name by which the user prefers to be known.

Product ID. The product ID of the user-defined field. If the value is *NONE, there is no product ID.

Print cover page. Whether a cover page is printed when the user's mail is printed. The values are 0 for no and 1 for yes.

Print personal mail. This field is used only if the user is an indirect user. The value specifies whether to print the mail that can be accessed only by the receiver, but not by someone working on behalf of the receiver. When mail is sent, it can be assigned the classification personal. The values are 0 for no and 1 for yes.

Reports to department. The department to which this department reports.

Reserved. An ignored field.

System name/group. An IBM-supplied name that uniquely identifies the system. It is used as a network value for certain communications applications such as APPC.

Telephone number 1. The telephone number of the user's office or business, or telephone numbers that are significant to the user. The most important number should be listed on the first line because only the first line is displayed when you use the search directory function.

Telephone number 2. The second line for telephone numbers.

Text. Any additional information that describes the entry.

Title. A department title that further describes the department name.

User-defined fields. Fields that are defined on the system by the Change System Directory Attributes (CHGSYSDIRA) command. The value of these user-defined fields can then be filled in on the directory entry for each user. The field name, product ID and field value are passed in the user-defined fields array. The product ID will be blank if the product ID does not exist.

User ID/address. A two-part network name used in the system distribution directory and in the office applications to uniquely identify a user and to send electronic mail.

User profile. The user profile name, if any, associated with a user ID and address.

X.400 administration domain. The administration management domain part of the X.400 originator/recipient (O/R) name. An administration management domain is a management domain that is administered by a public organization, such as a national Post Telephone and Telegraph Administration (PTT). A management domain is a set of message transfer agents (MTAs) and user agents (UAs) that comprise a message handling system.

X.400 country or region. The country or region part of the X.400 originator/recipient (O/R) name.

X.400 domain attribute types 1 through 4. The type of a domain-defined attribute for this object. The domain-defined attribute is not defined by X.400 standards but is allowed in the X.400 originator/recipient (O/R) name to accommodate values of existing systems sending messages.

X.400 domain attribute values 1 through 4. The code immediately following the attribute type that specifies a particular property from the set defined by the attribute type.

X.400 generation qualifier. The generation qualifier part of the X.400 originator/recipient (O/R) name. For example, the generation qualifier in the name John R. Smith, III, is III. If you specify a generation qualifier, you must specify an X.400 surname.

X.400 given name. The user first name, or given name, part of the X.400 originator/recipient (O/R) name. The default for the given name is the equivalent of the first name. If you specify a given name, you must specify an X.400 surname.

X.400 initials. The first and middle initials of the X.400 originator/recipient (O/R) name. For example, the initials for John Henry Smith are JH. If you specify initials, you must specify a surname.

X.400 organization. The organization name part of the X.400 originator/recipient (O/R) name.

X.400 organization units 1 through 4. The organization-defined unit part of the X.400 originator/recipient (O/R) name.

X.400 private domain. The private management domain part of the X.400 originator/recipient (O/R) name. A private management domain is a management domain that is administered by a private company or a noncommercial organization.

X.400 surname. The user last name, or surname, part of the X.400 originator/recipient (O/R) name.

Error Messages

Message ID	Error Message Text
CPF89A3 E	Operation not successful due to authority reasons.
CPF89A4 E	Operation not successful due to data validation reasons.

Exit Program Introduced: V3R6

Directory Search Exit Program

Required Parameter Group:

1	Function being requested	Input	Char(10)
2	Maximum entries for addressees	Input	Binary(4)
3	Maximum entries for copy list	Input	Binary(4)
4	Number of array elements	Output	Binary(4)
5	Array	Output	Char(*)
6	Maximum entries for blind copy list	Input	Binary(4)

```
QSYSINC Member Name:  EOKDRSH1
```

The Directory Search exit program allows the administrator to define a customized search of directory data. The exit will perform one of three functions: display, select, or optional select. In this way, a user-customized search is provided and this search is integrated with OfficeVision programs and the system distribution directory.

Use the Change System Directory Attributes (CHGSYSDIRA) command to specify the Directory Search exit program name in the SCHPGM parameter. When F10 is pressed on the Search System Directory display, the Directory Search exit program is called.

Required Parameter Group

Function being requested

INPUT; CHAR(10)

The type of search operation that was requested.

**DISPLAY*	Search and display directory information.
**SELECT*	Search and select user IDs. The output includes user IDs and addresses selected as a result of a search function.
**OPTSELECT*	Search and select user IDs for addressees and copy list.

Maximum entries for addressees

INPUT; BINARY(4)

The maximum number of entries that can be selected for the addressees of the distribution. The maximum number allowable is 100.

This parameter is used with the *SELECT and *OPTSELECT values of the function being requested parameter. If the function is *DISPLAY, this value is 0 indicating no output is returned.

Maximum entries for copy list

INPUT; BINARY(4)

The maximum number of entries that can be selected to be copied on the distribution. This is the maximum number of entries that can be returned in the array parameter with the value of 2 in select option used field. The maximum number allowable is 100.

This value is used with the *OPTSELECT value of the function being requested parameter. If the function is either *SELECT or *DISPLAY, this value is 0 indicating no output is returned.

Number of array elements

OUTPUT; BINARY(4)

For the values *SELECT and *OPTSELECT, the number of users that is returned. The maximum value is the total of the maximum entries for addressees parameter plus the maximum entries for copy list parameter.

Array

OUTPUT; CHAR(*)

For the values *SELECT and *OPTSELECT, the array of the user IDs and addresses selected from the search function. For the format of this character parameter follows.

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	CHAR(1)	Select option used
1	1	CHAR(8)	User ID
9	9	CHAR(8)	User address
17	11	CHAR(50)	User's full name or description

Maximum entries for blind copy list

INPUT; BINARY(4)

The maximum number of entries that can be selected to be blind copied on the distribution. This is the maximum number of entries that can be returned in the array parameter with the value of 4 in select option used field. The maximum number allowable is 100.

This value is used with the *OPTSELECT value of the function being requested parameter. If the function is either *SELECT or *DISPLAY, this value is 0 which indicates that no output is returned.

Field Descriptions

Select option used. The number of the option you want to use. The following are the possible values:

1	Addressee list
2	Carbon copy list (used only with *OPTSELECT)
4	Blind copy list (used only with *OPTSELECT)

User address. The second part of a two-part network name used in the system distribution directory and in the office applications to uniquely identify a user and to send electronic mail.

User ID. The first part of a two-part network name used in the system distribution directory and in the office applications to uniquely identify a user.

User's full name or description. The user's full name or any description up to 50 characters.

Exit Program Introduced: V2R2

Directory Supplier Exit Program

Required Parameter Group:

1	Function being requested	Input	Char(10)
2	Directory information format	Input	Char(10)
3	Owning system name	Input	Char(8)
4	User making request	Input	Char(10)
5	System making request	Input	Char(8)
6	Length of directory information	Input	Binary(4)
7	Directory information	Input	Char(*)
8	User exit program type	Input	Char(10)

```
QSYSINC Member Name:  EOKDRSP

Exit Point Name:  QIBM_QOK_SUPPLIER

Exit Point Format Name:  SUPL0100
```

The supplier program allows the administrator to decide whether operations (add, change, or delete) for directory entries, departments, and locations should be shadowed to collector systems. If a user exit program is defined, it is called before any changes are shadowed to a collector system. The supplier program is specified on the SUPPGM parameter of the Change System Directory Attributes (CHGSYSDIRA) command, or through the registration facility for common exit programs.

Exit programs that have been registered through the registration facility can be viewed using the Work with Registration Information (WRKREGINF) command.

During directory shadowing, the supplier program is given all information known about the changes to the directory entries, departments, and locations. The exit program returns to the directory service an indication as to whether the add, change, or delete operation should be supplied to the collecting system.

The supplier program can reduce the amount of processing on the network, as opposed to the verification maintenance program. The supplier program can decide not to supply changes to the collecting system. However, the verification maintenance program can only decide not to apply changes that have already been supplied to the collecting system.

The supplier criteria should be consistent on all your systems to help reduce the amount of processing on the network.

Required Parameter Group

Function being requested

INPUT; CHAR(10)

The type of operation that the user is requesting to do to the directory information that is described by the other parameters.

**ADD*	Information is being added.

**ADDDSC*	User description is being added.
**CHG*	Information is being changed. All fields in the directory information that are not changed will be X'00' except for the first field of every table. That field indicates what directory entry, department, or location is being changed.
**DLT*	Information is being deleted.
**DLTDSC*	User description is being deleted.

Directory information format

INPUT; CHAR(10)

The format of the directory information that is being worked with. The information is provided in the directory information parameter. The valid formats are:

SUPP0100	Directory entry (See SUPP0100 Format.)
SUPP0200	Department entry (See SUPP0200 Format.)
SUPP0300	Location entry (See SUPP0300 Format.)

These formats have the same layout as the CHKP0100, CHKP0200, and CHKP0300 formats used for the Directory Maintenance exit program. This allows a single program to be used as both a supplier program and a verification maintenance program.

Owning system name

INPUT; CHAR(8)

The name of the system that "owns" the directory entry, department, or location that is being worked with. The owning system is the system that originally added the data to the network.

**LOCAL*	The entry is owned by the local system.

User making request

INPUT; CHAR(10)

The user profile name of the user that is doing the request. When using the shadowing function, this is the user that originated the change.

System making request

INPUT; CHAR(8)

The system from which the request is coming. When using the shadowing function, this is the system that originated the change.

Length of directory information

INPUT; BINARY(4)

The length of the directory information in the directory information parameter. The length depends on the directory information format. Each format has a different (but fixed) length as shown in specific format tables.

Directory information

INPUT; CHAR(*)

The directory information that is associated with the directory entry, department, or location that the request is made against. For the format of this character parameter, refer to the specific format table (SUPP0100 Format, SUPP0200 Format , or SUPP0300 Format) and to the Field Descriptions.

User exit program type

INPUT; CHAR(10)

The user exit program type that is associated with the call of the Directory Supplier exit program. This parameter is provided so that a single program can be used as both a supplier program and a verification program. This parameter is always set to *SUPPGM.

The Directory Supplier exit program will support a required parameter of 8 or 9 parameters. This is so the same exit program can be used for the Directory Supplier exit program or the verification maintenance exit program. To use a Directory Supplier exit program with 9 parameters, the first 8 parameters should be defined as documented for this exit program. The ninth parameter should be defined as documented for the ninth parameter for the verification maintenance exit program. When the Directory Supplier exit program is called with 9 parameters, the ninth parameter is ignored.

See Directory Maintenance Exit Program for more information on the verification maintenance exit program.

Rejecting a Directory Shadowing Record

The user-written exit program may reject requests to supply changes to a collector system. To do so, the exit program must signal a specific program message to the directory services module that called it, and then return. A directory shadowing operation is rejected based on the restrictions that have been identified. For example, if the exit program allows only five of ten systems in an advanced program-to-program communications (APPC) network to collect directory information, then directory shadowing requests by all other systems are rejected.

To allow a directory shadowing record to be supplied, the exit program only returns to the program that called it. Two program messages have been defined for the purpose of rejecting directory shadowing records. The messages are:

CPF89B6	Directory information not shadowed for authority reasons.
CPF89B8	Directory information not shadowed for data validation reasons.

Note: The message must be signalled as an escape message. A diagnostic or informational message can be signalled before the escape message to give additional information about the error.

You may provide an optional data structure with message variable substitution text. This will help clarify the reasons for the rejection to the users. The optional data structure is:

CHAR(10)	The profile name of the user who requested directory shadowing. This information is from the user making request parameter.
CHAR(8)	The system name of the user who requested directory shadowing. You can get this from the system making request parameter.
CHAR(120)	A description of the reason the exit program is rejecting the request.

SUPP0100 Format

Offset		Type	Field
Dec	Hex		
Note: The following fields are in code page 500 and character set 697.			
0	0	CHAR(16)	User ID/address
16	10	CHAR(16)	System name/group
32	20	CHAR(10)	User profile
42	2A	CHAR(47)	Network user ID
89	59	CHAR(16)	New user ID/address
105	69	CHAR(16)	Old user to forward from user ID/address
121	79	CHAR(1)	Indirect user
122	7A	CHAR(1)	Print personal mail
123	7B	CHAR(3)	Reserved
Note: The character sets and code pages immediately follow the individual fields in the list below.			
126	7E	CHAR(50)	Description
176	B0	BINARY(4)	Character set
180	B4	BINARY(4)	Code page
184	B8	CHAR(40)	Last name
224	E0	BINARY(4)	Character set
228	E4	BINARY(4)	Code page
232	E8	CHAR(20)	First name
252	FC	BINARY(4)	Character set
256	100	BINARY(4)	Code page
260	104	CHAR(20)	Middle name
280	118	BINARY(4)	Character set
284	11C	BINARY(4)	Code page
288	120	CHAR(20)	Preferred name
308	134	BINARY(4)	Character set
312	138	BINARY(4)	Code page
316	13C	CHAR(2)	Reserved
318	13E	CHAR(50)	Full name
368	170	BINARY(4)	Character set
372	174	BINARY(4)	Code page
376	178	CHAR(2)	Reserved
378	17A	CHAR(10)	Department
388	184	BINARY(4)	Character set
392	188	BINARY(4)	Code page
396	18C	CHAR(2)	Reserved
398	18E	CHAR(50)	Job title
448	1C0	BINARY(4)	Character set

452	1C4	BINARY(4)	Code page
456	1C8	CHAR(2)	Reserved
458	1CA	CHAR(50)	Company
508	1FC	BINARY(4)	Character set
512	200	BINARY(4)	Code page
516	204	CHAR(2)	Reserved
518	206	CHAR(26)	Telephone number 1
544	220	BINARY(4)	Character set
548	224	BINARY(4)	Code page
552	228	CHAR(2)	Reserved
554	22A	CHAR(26)	Telephone number 2
580	244	BINARY(4)	Character set
584	248	BINARY(4)	Code page
588	24C	CHAR(40)	Location
628	274	BINARY(4)	Character set
632	278	BINARY(4)	Code page
636	27C	CHAR(20)	Building
656	290	BINARY(4)	Character set
660	294	BINARY(4)	Code page
664	298	CHAR(16)	Office
680	2A8	BINARY(4)	Character set
684	2AC	BINARY(4)	Code page
688	2B0	CHAR(40)	Mailing address line 1
728	2D8	BINARY(4)	Character set
732	2DC	BINARY(4)	Code page
736	2E0	CHAR(40)	Mailing address line 2
776	308	BINARY(4)	Character set
780	30C	BINARY(4)	Code page
784	310	CHAR(40)	Mailing address line 3
824	338	BINARY(4)	Character set
828	33C	BINARY(4)	Code page
832	340	CHAR(40)	Mailing address line 4
872	368	BINARY(4)	Character set
876	36C	BINARY(4)	Code page
880	370	CHAR(2)	Reserved
882	372	CHAR(50)	Text
932	3A4	BINARY(4)	Character set
936	3A8	BINARY(4)	Code page
940	3AC	CHAR(1)	Print cover page
941	3AD	CHAR(1)	Mail notification
Note: The following X.400 fields are in the character set and code page as defined by 1984 X.400 standards.			

942	3AE	CHAR(3)	X.400 country or region
945	3B1	CHAR(16)	X.400 administration domain
961	3C1	CHAR(16)	X.400 private domain
977	3D1	CHAR(64)	X.400 organization
1041	411	CHAR(40)	X.400 surname
1081	439	CHAR(16)	X.400 given name
1097	449	CHAR(5)	X.400 initials
1102	44E	CHAR(3)	X.400 generation qualifier
1105	451	CHAR(32)	X.400 organization unit 1
1137	471	CHAR(32)	X.400 organization unit 2
1169	491	CHAR(32)	X.400 organization unit 3
1201	4B1	CHAR(32)	X.400 organization unit 4
1233	4D1	CHAR(8)	X.400 domain attribute type 1
1241	4D9	CHAR(128)	X.400 domain attribute value 1
1369	559	CHAR(8)	X.400 domain attribute type 2
1377	561	CHAR(128)	X.400 domain attribute value 2
1505	5E1	CHAR(8)	X.400 domain attribute type 3
1513	5E9	CHAR(128)	X.400 domain attribute value 3
1641	669	CHAR(8)	X.400 domain attribute type 4
1649	671	CHAR(128)	X.400 domain attribute value 4
1777	6F1	CHAR(3)	Reserved
1780	6F4	CHAR(32)	Fax telephone number
1812	714	BINARY(4)	Character set
1816	718	BINARY(4)	Code page
1820	71C	CHAR(17)	Mail service level
1837	72D	CHAR(29)	Preferred address
1866	74A	CHAR(255)	cc:Mail address
2121	849	CHAR(126)	cc:Mail comment
2247	8C7	CHAR(1)	Allow synchronization
2248	8C8	BINARY(4)	Offset to user-defined fields array
2252	8CC	BINARY(4)	Number of elements (fields) in user-defined fields array
2256	8D0	CHAR(10)	DLO owner
Note: All fields that are not changed will be X'00' except for the user ID/address field.			

Array for User-Defined Fields

The following table is the array for the user-defined fields, format SUPP0100.

Offset			
Dec	**Hex**	**Type**	**Field**

0	0	BINARY(4)	Displacement to next user-defined field element in this array
4	4	CHAR(10)	Field name
14	E	CHAR(7)	Product ID
21	15	CHAR(3)	Reserved
24	18	BINARY(4)	Character set
28	1C	BINARY(4)	Code page
32	20	BINARY(4)	Length of field value returned
36	24	CHAR(*)	Field value

SUPP0200 Format

Offset			
Dec	**Hex**	**Type**	**Field**
Note: The character sets and code pages immediately follow the individual fields in the list below.			
0	0	CHAR(2)	Reserved
2	2	CHAR(10)	Department
12	C	BINARY(4)	Character set
16	10	BINARY(4)	Code page
20	14	CHAR(2)	Reserved
22	16	CHAR(50)	Title
72	48	BINARY(4)	Character set
76	4C	BINARY(4)	Code page
80	50	CHAR(2)	Reserved
82	52	CHAR(10)	Reports to department
92	5C	BINARY(4)	Character set
96	60	BINARY(4)	Code page
Note: The following field is in code page 500 and character set 697.			
100	64	CHAR(16)	Manager user ID/address
116	74	CHAR(2)	Reserved
118	76	CHAR(10)	Old department
128	80	BINARY(4)	Character set
132	84	BINARY(4)	Code page
Note: All fields that are not changed will be X'00' except for the department field and its corresponding character set and code page fields.			

SUPP0300 Format

Offset			
Dec	**Hex**	**Type**	**Field**
Note: The character sets and code pages immediately follow the individual fields in the list below.			
0	0	CHAR(40)	Location
40	28	BINARY(4)	Character set
44	2C	BINARY(4)	Code page
48	30	CHAR(2)	Reserved
50	32	CHAR(30)	Location line 1
80	50	BINARY(4)	Character set
84	54	BINARY(4)	Code page
88	58	CHAR(2)	Reserved
90	5A	CHAR(30)	Location line 2
120	78	BINARY(4)	Character set
124	7C	BINARY(4)	Code page
128	80	CHAR(2)	Reserved
130	82	CHAR(30)	Location line 3
160	A0	BINARY(4)	Character set
164	A4	BINARY(4)	Code page
168	A8	CHAR(2)	Reserved
170	AA	CHAR(30)	Location line 4
200	C8	BINARY(4)	Character set
204	CC	BINARY(4)	Code page
208	D0	CHAR(2)	Reserved
210	D2	CHAR(30)	Location line 5
240	F0	BINARY(4)	Character set
244	F4	BINARY(4)	Code page
248	F8	CHAR(2)	Reserved
250	FA	CHAR(30)	Location line 6
280	118	BINARY(4)	Character set
284	11C	BINARY(4)	Code page
288	120	CHAR(40)	Location changed to
328	148	BINARY(4)	Character set
332	14C	BINARY(4)	Code page
336	150	CHAR(40)	Old location
376	178	BINARY(4)	Character set
380	17C	BINARY(4)	Code page
Note: All fields that are not changed will be X'00' except for the location field and its corresponding character set and code page fields.			

Field Descriptions

Allow synchronization. Whether the directory entries should be synchronized with directories other than the system distribution directory. The values are 0 for no and 1 for yes.

Building. The name or number that identifies the user's building.

cc:Mail address. The cc:Mail address for a user. This field has a maximum of 126 characters, or 255 if the cc:Mail address contains both a remote post office name and an alias name.

cc:Mail comment. The cc:Mail comment for this user.

Character set. The character identifier (graphic character set) that was used by the work station to enter the data for the field.

Code page. The value specified on this parameter is used to instruct the printer device to interpret the hexadecimal byte string to print the same characters that were intended when the text was created.

Company. The name of the company for whom the user works.

Department. The name or number that identifies the user's department.

Description. The description associated with the user ID. One entry in the directory can have several different descriptions.

Displacement to next user-defined field element in this array. The displacement, in bytes, to the next user-defined field. Use this value to increment the pointer to get to the next user-defined field.

DLO owner. A special value indicating whether the user profile or the group profile will be assigned ownership of newly created document library objects (DLOs) associated with this directory entry.

Fax telephone number. The facsimile telephone number.

Field name. The user-defined field name.

Field value. The value of the user-defined field. The maximum is 512 bytes.

First name. The user's first name or given name.

Full name. The user's full name as it appears when a directory is viewed or searched.

Indirect user. A user enrolled in the system distribution directory who receives mail but never signs on to view it. An indirect user receives printed mail only. The values are 0 for no and 1 for yes.

Job title. The title of the user's occupation.

Last name. The user's last name.

Length of field value returned. The length of the user-defined field value. If the value is 0, there is no data in that field for this user.

Location. The location of the business or system. Some examples of location are city, state, or street address.

Location changed to. The new location value after combining locations.

Location lines 1 through 6. The location of the business or system. These fields further describe a location name. For example, the field may contain the general mailing address for the location.

Mail notification. Whether or not the user wants to be notified when mail arrives.

The user can specify these values:

Blank	Notified for priority or personal mail and messages
1	Notified for priority or personal mail and messages
2	Notified for only priority or personal mail
3	Notified for messages only
4	Notified for all mail
0	Notified for no mail

Mail service level. A 17-byte field where the first 10 bytes are the field name and the last 7 bytes are the product ID. The values for the mail service level can be:

**USRIDX*	User index
**SYSMS*	System message store
**DOMINO*	Lotus Domino mail database
A user-defined field name and product ID	The product ID will be blank if the product ID does not exist.

Mailing address lines 1 through 4. The address of the user.

Manager user ID/address. The department manager's user ID and address.

Middle name. The user's middle name.

Network user ID. A unique value associated with each user in the Enterprise Address Book. For example, the value could be the user ID/address, the social security number, or the employee number.

New user ID/address. The new user ID and address used during the rename operation.

Office. The name or number that identifies the user's office.

Old department. The previous department value before being changed.

Old location. The previous location value before being changed.

Old user to forward from user ID/address. This field shows the previous user ID and address from which the user forwards mail.

Preferred address. A 29-byte field where the first 10 bytes are the field name, the second 7 bytes are the product ID, the third 4 bytes are reserved, and the last 8 bytes are the address type name.

The special values of the field name in preferred address can be:

**USRID*	User ID/address
**ORNAME*	X.400 O/R name
**SMTP*	Simple Mail Transfer Protocol (SMTP) name
A user-defined or IBM-defined field	The product ID is *IBM for IBM-defined fields. The product ID will be blank if the product ID does not exist.

Preferred name. The name by which the user prefers to be known.

Print cover page. Whether a cover page is printed when the user's mail is printed. The values are 0 for no and 1 for yes.

Print personal mail. Whether to print the mail that can be accessed only by the receiver, but not by someone working on behalf of the receiver. When mail is sent, it can be assigned the classification of personal. This field is used only if the user is an indirect user. The values are 0 for No and 1 for Yes.

Product ID. The product ID of the user-defined field. If the value is *NONE, there is no product ID associated with the field.

Reports to department. The department to which this department reports.

Reserved. An ignored field.

System name/group. An IBM-supplied name that uniquely identifies the system. It is used as a network value for certain communications applications such as APPC.

Telephone number 1. The telephone number of the user's office or business, or telephone numbers that are significant to the user. The most important number should be listed on the first line because only the first line is displayed when you use the search directory function.

Telephone number 2. The second line for telephone numbers.

Text. Any additional information that describes the entry.

Title. A department title that further describes the department name.

User-defined fields. Fields that are defined on the system by the Change System Directory Attributes (CHGSYSDIRA) command. The value of these user-defined fields can then be filled in on the directory entry for each user. The field name, product ID, and the value is passed in the user-defined field array. The product ID will be blank if the product ID does not exist.

User ID/address. A two-part network name used in the system distribution directory and in the office applications to uniquely identify a user and to send electronic mail.

User profile. The user profile name, if any, associated with a user ID and address.

X.400 administration domain. The administration management domain part of the X.400 originator/recipient (O/R) name. An administration management domain is a management domain that is administered by a public organization, such as a national Post Telephone and Telegraph Administration (PTT). A management domain is a set of message transfer agents (MTAs) and user agents (UAs) that comprise a message handling system.

X.400 country or region. The country or region part of the X.400 originator/recipient (O/R) name.

X.400 domain attribute types 1 through 4. The type of a domain-defined attribute for this object. The domain-defined attribute is not defined by X.400 standards but is allowed in the X.400 originator/recipient (O/R) name to accommodate values of existing systems sending messages.

X.400 domain attribute values 1 through 4. The code immediately following the attribute type that specifies a particular property from the set defined by the attribute type.

X.400 generation qualifier. The generation qualifier part of the X.400 originator/recipient (O/R) name. For example, the generation qualifier in the name John R. Smith, III, is III. If you specify a generation qualifier, you must specify an X.400 surname.

X.400 given name. The user first name, or given name, part of the X.400 originator/recipient (O/R) name. The default for the given name is the equivalent of the first name. If you specify a given name, you must specify an X.400 surname.

X.400 initials. The first and middle initials of the X.400 originator/recipient (O/R) name. For example, the initials for John Henry Smith are JH. If you specify initials, you must specify a surname.

X.400 organization. The organization name part of the X.400 originator/recipient (O/R) name.

X.400 organization units 1 through 4. The organization-defined unit part of the X.400 originator/recipient (O/R) name.

X.400 private domain. The private management domain part of the X.400 originator/recipient (O/R) name. A private management domain is a management domain that is administered by a private company or a noncommercial organization.

X.400 surname. The user last name, or surname, part of the X.400 originator/recipient (O/R) name.

Error Messages

Message ID	Error Message Text
CPF89B6 E	Directory information not shadowed for authority reasons.
CPF89B8 E	Directory information not shadowed for data validation reasons.

Exit Program Introduced: V2R3

Document Conversion Exit Program

Required Parameter Group:

1	Input document name	Input	Char(12)
2	Input folder name	Input	Char(63)
3	Input document type	Input	Binary(4)
4	Output document name	Input	Char(12)
5	Output folder name	Input	Char(63)
6	Output document type	Input	Binary(4)
7	Function code	Input	Char(1)
8	Conversion existence indicator	Output	Char(1)

The Document Conversion exit program allows other document conversions to be called when a request is made for the OfficeVision program to process a document type that it does not support. The OS/400 and OfficeVision programs use document conversions when opening documents.

Program Registration

To register a user exit program, use the Change Office Program (QOGCHGOE) API.

Required Parameter Group

Input document name

INPUT; CHAR(12)

The name of the document the function is to be performed against.

Input folder name

INPUT; CHAR(63)

The folder in which the document is to be found.

Input document type

INPUT; BINARY(4)

The DIA document type ID. You can display a list of document types defined with the Work with Document Types (WRKDOCTYP) command. The value must be in the range of 1 through 65535.

Output document name

INPUT; CHAR(12)

The name of the output document.

Output folder name

INPUT; CHAR(63)

The folder in which the output document is to be placed.

Output document type

INPUT; BINARY(4)

The format of the document that is being worked with. The value must be in the range of 1 through 65535.

Function code

INPUT; CHAR(1)

Whether the exit is being called to check for the existence of a conversion or to perform the conversion.

0	A conversion is being requested.
1	An existence check is requested.

Conversion existence indicator

OUTPUT; CHAR(1)

Whether the requested conversion exists. This flag must be set for both conversion existence checks and conversion requests.

0	Conversion does not exist.
1	Conversion exists.

Use of Document Conversions by IBM Programs

Other document conversions will be called when you are opening documents and:

- Processing the following document commands and using the OfficeVision editor or print functions:

  CRTDOC

  EDTDOC

  DSPDOC

  PRTDOC

  CHKDOC

  MRGDOC

  PAGDOC

  ADDTXTIDXE
- Processing the following options from the Work with Documents in Folders display and using OfficeVision:

  Create

Revise

View

Print

Spell

Paginate

Print options

- Processing the following options from the Work with Documents in a Document List display:

  Revise

  View

  Print

- Processing the following options from the Work with Mail display and using OfficeVision:

  Revise a copy

  View

  Print

  Forward

  Reply

- Copying a document on a create request when using the OfficeVision editor.
- Recalling from or copying or moving to a notepad document within the OfficeVision editor.
- Using the GET function within the OfficeVision editor.
- Including a document with the .inc instruction in the OfficeVision print function.
- Accessing external footnote documents in the OfficeVision print function.
- Accessing a fill-in document (merge from a document) in the OfficeVision print function.
- Accessing shell documents QPROFNOT and QPROFDOC for converting incoming PROFS documents and notes.
- Printing mail for indirect office users.
- Accessing the shell note when creating a note.
- Accessing the output document for a copy graph or image request.

In each of these cases a conversion from the current format to RFTDCA or FFTDCA is requested. A second conversion from this resulting format to OS/400 format may also be performed using the IBM conversion programs.

Exit Program Introduced: V2R2

Top | Office APIs | APIs by category

Document Handling Exit Program

Required Parameter Group:			
1	Document name	Input	Char(12)
2	Folder name	Input	Char(63)
3	Document type	Input	Binary(4)
4	Function	Input	Char(26)
5	Function-specific information	Input	Char(*)
6	Exit processing indicator	Output	Char(4)
QSYSINC Member Name: EOGDOCH			

The Document Handling program allows other applications to be called in place of or in addition to the OfficeVision word processor. Requests are sent to the exit program. The exit program can choose to process the request or return it to the OfficeVision program for processing.

OfficeVision application enabler support assists users in integrating other office applications into OfficeVision. This is accomplished by creating command interfaces that define the applications to OfficeVision in terms of document types. Applications created this way can support the following functions:

Table 1. APIs Used to Register User Exit Applications

Function	API Used for Registration	Exit Point Name and Exit Point Format Name
ADDRESSING	Add Exit Program API	QIBM_QOE_OV_USR_SND, DOCI0900
ADDRCANCEL	Add Exit Program API	QIBM_QOE_OV_USR_SND, DOCI0900
CREATE	QOGCHGOE	
EDIT	QOGCHGOE	
FILLFORM	QOGCHGOE	
MAILEDIT	QOGCHGOE	
MAILVIEW	QOGCHGOE	
MAILFWD	QOGCHGOE	
MAILREPLY	QOGCHGOE	
MERGE	QOGCHGOE	
MERGEOPTS	QOGCHGOE	
PAGINATE	QOGCHGOE	
PRINT	QOGCHGOE	
PRINTOPTS	QOGCHGOE	
SEND	Add Exit Program API	QIBM_QOE_OV_USR_SND, DOCI0900
SPELLCHECK	QOGCHGOE	
VIEW	QOGCHGOE	

Note: For performance improvements, a local copy of the API information is cached on the first usage by a job. The cached API information is then available for reuse by the executing job. (For example, when a job needs information about a registered user exit program, the information is cached.) Therefore, the changes to the API may not take effect until after the jobs with the cached information are terminated. This means that a user may need to sign off then sign on again before the API changes become effective.

The sign off process is an example of a job termination. The sign on process starts a new job.

Required Parameter Group

Document name

INPUT; CHAR(12)

The name of the document on which the function is performed.

Folder name

INPUT; CHAR(63)

The folder in which the document is located.

Document type

INPUT; BINARY(4)

The DIA document type ID. You can display a list of document types defined with the Work with Document Types (WRKDOCTYP) command. The value must be from 1 to 65535.

Function

INPUT; CHAR(26)

The first 10 characters is the type of operation that the user is requesting for this document. The next 16 characters contain the address of the current work on behalf of user. If there is no work on behalf of user, it will contain spaces.

Refer to the DOCI0100 Format for print function requests.

Refer to the DOCI0200 Format for merge function requests.

Refer to the DOCI0300 Format for spell function requests.

Refer to the DOCI0400 Format for mail function requests.

Refer to the DOCI0500 Format for edit function requests.

Refer to the DOCI0600 Format for create function requests.

Refer to the DOCI0700 Format for fill form function requests.

Refer to the DOCI0800 Format for mail view function requests.

Refer to the DOCI0900 Format for send, addressing, and cancel user application addressing function requests.

Function-specific information

INPUT; CHAR(*)

Additional input that varies by function. See Table 2 for more detailed information.

Note: Function-specific information that is not specified and for which there is no value stored in the document will be passed to the exit program as blanks.

Exit processing indicator

OUTPUT; CHAR(4)

The additional processing that the IBM programs should perform on return from the document handling program. Any code returned, when the function requested does not support it, will be treated as a return code of 0000.

0000 No additional processing required for this request. (Processed like the enter key from the exit.)

0001 Request has been processed, and the user requested to return. The IBM programs process as if F12 were pressed.

0002 Request has been processed, and the user requested to return. The IBM programs process as if F3 were pressed.

0007 Request OfficeVision to delete the document from the mail log (only valid on VIEWMAIL requests).

0008 Request has been processed and the user requested that the item be sent. Only the MailEdit, MailForward, MailReply, Addressing, and Addcancel functions are valid.

0010 The OfficeVision program will process the requester as if the user exit had not been called.

- ❍ For the user application send exit program (ADDRESSING, ADDRCANCEL, and SEND functions), additional processing will occur.

  - ■ If a higher numbered registered user exit program has been defined, that exit program will be called to process the request as if the previous user exit program had not been called.

  - ■ If 0010 is returned from the highest numbered registered exit program, OfficeVision will treat the request as an exit processing indicator of 0000.

- ❍ Considerations for a mail document:

  When working with a mail document, a copy of the original mail document is made. This is done to preserve the integrity of the original mailed document. This copied document is the document that is passed to the application user exit program.

  If a user exit program decides to use the 0010 exit processing indicator on exit (that is, let OfficeVision handle this document), then the application enabler will check to see if a higher numbered user exit program is registered. If so, this second user exit program will get passed to the copied document (this is the copied document that the first user exit program had access to and had the opportunity to modify or transform the data stream content).

  Now if the second user exit program decides to use the 0010 exit processing indicator on exit (and this is the highest number user exit program), the application enabler will return to mail with the request to let OfficeVision process the document.

It is the responsibility of the user exit programs to maintain the integrity of the copied document. It is possible that one of the user exit programs may have modified the document before the user exit program returns (for example, the user exit program had the opportunity to modify any part of the document or perhaps it may have transformed it to another data stream content).

The customer installation is responsible for maintaining the integrity of this copied document. If it is important that the exact document is kept upon return from the user exit program, perhaps the user exit program must make a copy before it calls the application program to process the document. It is the responsibility of the user exit program to handle deleting the copied document that it created.

0011 User is requesting that the document be filed locally. If any function other than MailView is used, it is treated as a 0000 return.

0012 User is requesting that the document be filed remotely. If any function other than MailView is used, it is treated as a 0000 return.

0013 User is requesting that the document be forwarded. If any function other than MailView is used, it is treated as a 0000 return.

0014 User is requesting a reply to the document. If any function other than MailView is used, it is treated as a 0000 return.

Table 2 shows when the exit program will be called for each of the document functions.

Table 2. Document Handling Function Requests

Function	**Description**	**Where Called From**
'ADDRESSING'	User application addressing(DOCI0900 format)	● *Send Note* display; F15 ● *Send Document* display; F15 ● *Send Form* display; F15 ● *Forward a Mail Item* display; F15 ● *Reply to a Mail Item* display; F15 ● SNDDOC command; F15

'ADDRCANCEL'	Cancel user application addressing (DOCI0900 format)	• *Send Note* display; F12; F3 • *Send Document* display; F12; F3 • *Send Form* display; F12; F3 • *Forward a Mail Item* display; F12; F3 • *Reply to a Mail Item* display; F12; F3 • SNDDOC command; F12; F3
'CREATE'	Create document (DOCI0600 format)	• CRTDOC command • *Work with Documents in Folders* display; option 1 **Note:** The exit program is called after the Enter key or F10 is pressed on the Create Document Details display, when details are used.
'VIEW'	View document (No function specific format information)	• DSPDOC command • MERGE request with DSPOPT(*VIEW) • *Work with Documents in Folders* display; option 5 • *Work with Documents in a List* display; option 5
'EDIT'	Edit document and revise a copy of a mail item (DOCI0500 format)	• EDTDOC command • MERGE request with DSPOPT(*EDIT) • *Work with Documents in Folders* display; option 2 • *Work with Documents in a List* display; option 2 • *Work with Mail* display; option 2

'FILLFORM'	Fill Form Document (complete a form using the fill form editor of OfficeVision for AS/400) (DOCI0700 format)	• FILLFORM command • MRGDOC command; DSPOPT is *FILLFORM • *Work with Documents in a List* display; option 15 **Note:** The exit program is called after the Enter key or F10 is pressed on the Create Document Details display, when details are used.
'MAILVIEW'	View mail item (DOCI0800 format)	• *Work with Mail* display; option 5
'MAILEDIT'	Create a note (DOCI0400 format)	• *OfficeVision* main menu; option 4 **Note:** The exit program is called after F6 (Type Note) is pressed on the *Send Note* display.
'MAILFWD'	Forward a note (DOCI0400 format)	• *Work with Mail* display; option 10 **Note:** The exit program is called after F6 (Type Note) is pressed on the *Forward Mail* display.
'MAILREPLY'	Reply to a note (DOCI0400 format)	• *Work with Mail* display; option 11 **Note:** The exit program is called after F6 (Type Note) is pressed on the *Reply to Mail* display.
'MERGE'	Merge document (DOCI0200 format)	• MRGDOC command; display merge options is *NO
'MERGEOPTS'	Merge document with options (DOCI0200 format)	• MRGDOC command; display merge options is *YES
'PRINT'	Print request (DOCI0100 format)	• PRTDOC command; display print options is *NO • *Work with Documents in Folders* display; option 6 • *Work with Documents in a List* display; option 6 • *Work with Mail* display; option 6

'PRINTOPTS'	Print with options (DOCI0100 format)	<ul><li>PRTDOC command; Display print options is *YES</li><li>*Work with Documents in Folders* display; option 9</li><li>*Work with Documents in a List* display; option 9</li><li>*Work with Mail* display; option 9</li></ul>
'PAGINATE'	Paginate document (DOCI0100 format)	<ul><li>PAGDOC command</li><li>*Work with Documents in Folders* display; option 13</li></ul>
'SEND'	User application send (DOCI0900 format)	If F15 processing was previously preformed: <ul><li>*Send Note* display; F10</li><li>*Send Document* display; F10</li><li>*Send Form* display; F10</li><li>*Forward a Mail Item* display; F10</li><li>*Reply to a Mail Item* display; F10</li><li>SNDDOC command; F10</li><li>F10 or F11 from an edit session</li></ul>
'SPELLCHECK'	Check the spelling accuracy or grade level of a document (DOCI0300 format)	<ul><li>CHKDOC command</li><li>*Work with Documents in Folders* display; option 11</li></ul>

DOCI0100 Format

This is the function-specific information for PRINT and PRINTOPTS requests. This format is filled out the same as it would have looked if the OfficeVision program were to process the request. This format merges the parameters specified on the PRTDOC command with the print options record. The *Field* column describes the associated PRTDOC parameter for each format.

The Field Descriptions have more detailed information about the fields for this format.

Offset		Type	Field
Dec	**Hex**		
0	0	CHAR(2)	Number of copies to print

2	2	CHAR(1)	Output device
3	3	CHAR(10)	Printer device name
13	D	CHAR(10)	Output queue name
23	17	CHAR(10)	Output queue library name
33	21	CHAR(10)	Output file name
43	2B	CHAR(10)	Form type
53	35	CHAR(10)	Printer file name
63	3F	CHAR(10)	Printer file library name
73	49	CHAR(10)	Device file name
83	53	CHAR(10)	Device library name
93	5D	CHAR(10)	Device member name
103	67	CHAR(1)	Delay printing
104	68	CHAR(1)	Draft spacing
105	69	CHAR(1)	Print line numbers
106	6A	CHAR(1)	Resolve instructions
107	6B	CHAR(1)	Large print
108	6C	BINARY(4)	Graphic character set
112	70	BINARY(4)	Code page
116	74	CHAR(1)	Print separator page
117	75	CHAR(1)	Adjust line endings
118	76	CHAR(1)	Adjust page endings
119	77	CHAR(1)	Allow widow lines
120	78	CHAR(2)	Additional spaces to left
122	7A	CHAR(1)	Print change symbols
123	7B	CHAR(5)	Change symbols to print
128	80	CHAR(1)	Print quality
129	81	CHAR(1)	Place on job queue
130	82	CHAR(1)	Send completion message
131	83	CHAR(10)	Job description name
141	8D	CHAR(10)	Job description library name
151	97	CHAR(1)	Cancel on error
152	98	CHAR(1)	Print error log
153	99	CHAR(10)	Error log form type
163	A3	CHAR(1)	Clear error log after printing
164	A4	CHAR(1)	Save resolved output
165	A5	CHAR(12)	Resolved output document
177	B1	CHAR(63)	Resolved output folder
240	F0	CHAR(1)	Multiple line report
241	F1	CHAR(1)	Print on both sides
242	F2	CHAR(1)	Automatic page binding
243	F3	CHAR(1)	Automatically shift margins to avoid truncating text

244	F4	CHAR(1)	Renumber system page numbers
245	F5	CHAR(7)	Print from page 1
252	FC	CHAR(8)	Print to page 1
260	104	CHAR(7)	Print from page 2
267	10B	CHAR(8)	Print to page 2
275	113	CHAR(7)	Print from page 3
282	11A	CHAR(8)	Print to page 3
290	122	CHAR(7)	Print from page 4
297	129	CHAR(8)	Print to page 4
305	131	CHAR(7)	Print from page 5
312	138	CHAR(8)	Print to page 5
320	140	CHAR(7)	Print from page 6
327	147	CHAR(8)	Print to page 6
335	14F	CHAR(7)	Print from page 7
342	156	CHAR(8)	Print to page 7
350	15E	CHAR(1)	Document is a label document
351	15F	CHAR(2)	Number of labels
353	161	CHAR(3)	Width of labels
356	164	CHAR(1)	Sheet-feed labels
357	165	CHAR(2)	Number of rows per sheet
359	167	CHAR(1)	Merge type
360	168	CHAR(10)	Query name
370	172	CHAR(10)	Query library name
380	17C	CHAR(12)	Data document name
392	188	CHAR(63)	Data folder name
455	1C7	CHAR(10)	Data file name
465	1D1	CHAR(10)	Data file library name
475	1DB	CHAR(10)	Data file member name

DOCI0200 Format

This is the structure that is used by this exit program when the function is for merge or merge with options. Note that the single-record merge is not supported.

The merge situations where the user specified DSPOPT(*VIEW) or DSPOPT(*EDIT) results in the exit program being called twice, once for the merge and then once for the subsequent view or edit. The *Field* column describes the associated MRGDOC parameter for each field.

The Field Descriptions have more detailed information about the fields for this format.

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	CHAR(12)	To document name

12	C	CHAR(63)	To folder name
75	4B	CHAR(1)	Replace document
76	4C	CHAR(1)	Place on job queue
77	4D	CHAR(1)	Send completion message
78	4E	CHAR(10)	Job description name
88	58	CHAR(10)	Job description library name
98	62	CHAR(1)	Merge type
99	63	CHAR(10)	Query name
109	6D	CHAR(10)	Query library name
119	77	CHAR(12)	Data document name
131	83	CHAR(63)	Data folder name
194	C2	CHAR(10)	Data file name
204	CC	CHAR(10)	Data file library name
214	D6	CHAR(10)	Data file member name
224	E0	CHAR(1)	Adjustment option
225	E1	CHAR(1)	Multiple line report
226	E2	CHAR(1)	Collect footnotes
227	E3	CHAR(258)	Reserved

DOCI0300 Format

The Field Descriptions explain the values to use for spell function requests.

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	CHAR(1)	Type of check
1	1	CHAR(7)	Beginning page number
8	8	CHAR(7)	Ending page number
15	F	CHAR(470)	Reserved

DOCI0400 Format

The Field Descriptions explain the values to use for mail function requests.

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	CHAR(12)	Mail reference document name
12	C	CHAR(63)	Mail reference folder name
75	4B	CHAR(1)	Attach mail reference
76	4C	CHAR(1)	Same note

77	4D	CHAR(408)	Reserved

DOCI0500 Format

The Field Descriptions explain the values to use for edit function requests.

Offset		Type	Field
Dec	Hex		
0	0	CHAR(1)	Show exit display
1	1	CHAR(484)	Reserved

DOCI0600 Format

The Field Descriptions explain the values to use for create function requests.

Offset		Type	Field
Dec	Hex		
0	0	CHAR(1)	Display exit display
1	1	CHAR(1)	Display document details display
2	2	CHAR(1)	Edit document
3	3	CHAR(482)	Reserved

DOCI0700 Format

The Field Descriptions explain the values to use for fill form function requests.

Offset		Type	Field
Dec	Hex		
0	0	CHAR(12)	Shell form
12	C	CHAR(63)	Shell folder
75	4B	CHAR(12)	Save form
87	57	CHAR(63)	Save folder
150	96	CHAR(1)	Replace form
151	97	CHAR(1)	Display status
152	98	CHAR(1)	Display exit
153	99	CHAR(1)	Allow refresh
154	9A	CHAR(10)	Dump data file
164	A4	CHAR(10)	File library name
174	AE	CHAR(10)	File member name

184	B8	CHAR(1)	Replace or add
185	B9	CHAR(1)	Output data on exit
186	BA	CHAR(299)	Reserved

DOCI0800 Format

The Field Descriptions explain the values to use for mail view function requests.

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	CHAR(8)	User ID (DEN)
8	8	CHAR(8)	User address (DGN)
16	10	CHAR(22)	Distribution ID
38	26	CHAR(447)	Reserved

DOCI0900 Format

The Field Descriptions explain the values to use for send and addressing function requests.

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	BINARY(4)	Size of format
4	4	BINARY(4)	Offset to recipient list
8	8	BINARY(4)	Number of addressees
12	C	BINARY(4)	Number of copy list recipients
16	10	BINARY(4)	Number of blind copy list recipients
20	14	CHAR(1)	Confirm on delivery
21	15	CHAR(1)	Log outgoing mail status
22	16	CHAR(1)	Sensitivity
23	17	CHAR(1)	Grade of delivery
24	18	CHAR(1)	Disclose recipients
25	19	CHAR(1)	Importance
26	1A	CHAR(1)	Allow alternate recipient
27	1B	CHAR(1)	Allow X.400 conversion
28	1C	CHAR(1)	Reply requested
29	1D	CHAR(7)	Reply requested date
36	24	CHAR(6)	Reply requested time
42	2A	CHAR(1)	Function
43	2B	CHAR(60)	Subject
103	67	BINARY(4)	Subject character set

107	6B	BINARY(4)	Subject code page
111	6F	CHAR(60)	Reference
171	AB	BINARY(4)	Reference character set
175	AF	BINARY(4)	Reference code page
179	B3	BINARY(4)	Memo slip character set
183	B7	BINARY(4)	Memo slip code page
187	BB	CHAR(30)	Action item text
217	D9	CHAR(202)	User's memo slip text
419	1A3	CHAR(13)	Reserved
Note: The following fields repeat as required.			
		CHAR(8)	User ID (DEN)
		CHAR(8)	User address (DGN)
Notes: 1. The user ID and user address fields are in the order of addressee list, copy list, and blind copy list. The number of users in an addressee list, a copy list, or a blind copy list do not affect the order. 2. All character fields are left-aligned and padded by blanks unless otherwise noted.			

Field Descriptions

Action item text. The actual text of the action item. As shipped, they are:

- For your information
- For your comments
- For your signature
- For your approval
- Please handle
- Please circulate
- Please see me
- Please prepare reply

Additional spaces to left. The number of spaces added to the left margin in your printed document. If the document is not printed on both sides of the paper, this is useful when you want to bind your document. Valid values are 0 through 99.

Adjust line endings. The value Y (yes) adjusts line endings in the printed document. The lines are adjusted according to what is specified on the Line Spacing/Justification Options display. This is useful when you print a document that has data merged into it, has instructions, has display attributes that do not print as spaces, or uses a proportionally spaced font.

The value N (no) means you do not want to adjust the line endings in the printed document.

Adjust page endings. The value Y (yes) adjusts page endings in the printed document. The pages are determined by what is specified for the *First typing line* and *Last typing line* prompts on the Page Layout/Paper Options display.

The value N (no) means you do not want to adjust the page endings in the printed document.

Adjustment option. The values indicate the adjustment option you want to use.

1	None
2	Lines (Adjusts line endings in the printed document.)
3	Pages and lines (Adjusts page endings and lines in the printed document.)

Allow alternate recipient. Whether the originator will allow applications to forward or delegate the mail to some alternate recipient.

0	No
1	Yes

Allow refresh. Whether the form refresh will be allowed from within the editor. The value 1 (yes) means the form refresh option will be allowed. The value 0 (no) means the refresh option will not be allowed.

Allow widow lines. The value Y (yes) means the page endings or column endings are determined by the exact number of lines per page as specified on the Page Layout/Paper Options display. Also, use Y (yes) for the *Adjust page endings* prompt. When you allow widows, a single line from a paragraph could be separated from the rest of the paragraph.

The value N (no) means you do not want to have page endings or column endings determined by the exact number of lines.

Allow X.400 conversion. Whether the originator will allow applications to make conversions to the distribution.

0	No
1	Yes

Attach mail reference. Whether to concatenate the original mail item with the forward or reply operation. For MAILREPLY, this is entered by the user on the Reply to Mail display. For MAILFWD, this will always be set to a value of yes. The values are yes or no.

Automatic page binding. You can choose to adjust margins for page binding, which is a way to print pages and adjust margins for the even pages so that they will line up with the odd pages when printing on both sides. The values are yes or no.

Automatically shift margins to avoid truncating text. You can automatically shift the margin so that as much text as possible is printed if the margin exceeds the paper edge. The values are yes or no.

Beginning page number. A number from 1.0 through 9999.99 or *FIRST or *LAST to specify the page on which you want spell checking to start.

Cancel on error. The value Y (yes) means you want the document to stop printing if an error is detected during printing. The error is listed in the error log with an error message stating that the job is canceled.

The value N (no) means you want the document to print even if an error is detected during printing.

Change symbols to print. Different revision symbols result in a document when the revision symbol is changed. If your document contains more than one revision symbol character, and you do not select which revision symbol characters you want to print, all revision symbol characters specified in your document will print.

Clear error log after printing. The value Y (yes) means you want previous errors removed from the existing error log page before new ones are added. This is useful if you want the error log to contain only the errors that occurred during one printing of the document.

The value N (no) means you want your errors added to the end of the existing error log page. This is useful if you want to keep a history of all the times your document printed.

Code page. A particular assignment of hexadecimal identifiers to graphic characters. If you want to specify a code page, type the graphic character-set and code-page combination. A code page is a particular assignment of hexadecimal identifiers to graphic characters. When both the graphic character-set ID and code-page ID are typed, they must be separated by a hyphen.

Collect footnotes. The value Y (yes) means you want the associated footnote text for all Footnote instructions found in the included text to become part of the document created during the merge function. All Footnote instructions are changed to refer to the correct system page.

The value N (no) means you want the associated footnote text for all Footnote instructions found in the included text to remain outside of the document created during the merge function.

Confirm on delivery. Whether to receive a notice in your outgoing mail log confirming that the note you sent has been delivered.

0	No
1	Yes

Data document name. The name of the document that contains the data to be merged.

The document name must be 1 to 12 characters in length.

Data file library name. The name of the library where the data file is located.

Data file member name. The name of the member that contains data.

Data file name. The name of the file to merge data from.

Data folder name. The name of the folder that contains the document that has the data to be merged.

Delay printing. The value Y (yes) delays printing your labels. The labels are held on the output queue where you can release them to print or delete them if you do not want them to print.

Device file name. The name of the file that contains the information about the device. Use the value 3 for file.

Device library name. A user-defined word that names a library. The value is *LIBL.

Device member name. The value *FILE for the member name, means the member with the same name as the file name will be used. The values are *FILE, *FIRST, and *LAST.

Disclose recipients. Whether your list of addressee and copy list recipients should be revealed to your recipients.

0	No
1	Yes

Display document details display. Whether you want the Document Details display shown. The values are Y (yes) or N (no).

Display exit. Whether the exit panel should be displayed.

0	No
1	Yes

Display exit display. Whether you want the exit display shown. The values are Y (yes) or N (no).

Display status. Whether status lines should be shown on the fill form display.

0	No
1	Yes

Document is a label document. The value (yes or no) identifies if the document is a label document. A label document contains the labels you are printing.

Draft spacing. The value Y (yes) doubles the spacing value in the *Line spacing* prompt on the Line Spacing/Justification Options display. For example, if the *Line spacing* prompt is 3 (Triple), the doubled spacing value is 6 and five blank lines are printed between each line of text in your document.

Dump data file. The name of the output file that will contain the form fill data.

Edit document. The values (yes or no) indicate whether you want to edit the document.

Ending page number. A number from 1.0 through 9999.99 or *FIRST or *LAST to specify the page on which you want spell checking to stop.

Error log form type. The forms type for the type of paper on which you want the error log to be printed. If you use the value *STD as the forms type, the error log will be printed on the paper specified in the printer file for the printer you selected. A printer file contains information controlling how your document is printed on a particular printer.

File library. The library to contain the dump data file.

File member. The member to contain the dump data. *FIRST is a special value indicating the first member of the file.

Form type. The host system form type for the forms control table (FCT) entry.

Function. Whether the user is forwarding or replying to a mail item or is sending a note or document.

0	User application send function requested from Send
1	User application send function requested from Forward/Reply

Grade of delivery. Allows you to specify the speed of the network for a note or message to be sent. The following values indicate the grade of delivery to use:

1 High

The distribution will be sent with the highest priority on the fastest distribution lines. It will appear highlighted in a user's mail log until the user takes action on it.

2 Normal

The note or message is sent through normal distribution channels.

3 Low

No special priorities or distribution methods will be used to send the note or message

Graphic character set. The graphic character-set ID that you want to print your job. A graphic character-set ID is an identifier used to specify a set of graphic characters in a code page. This identifier can be two 4-digit prompts separated by a blank or by a hyphen.

Importance. The importance the originator places on the content of the note. Values are as follows:

1	High
2	Normal
3	Low

Job description library name. The library name by specifying one of the following:

name	The name of the library that is storing your job description.
**LIBL*	If you use *LIBL, your library list is searched for the job description.

Job description name. The name of the job description that describes how the job is to be run.

Large print. The value Y (yes) prints your document with large print.

The value N (no) indicates that you do not want to print your document with large print.

Log outgoing mail status. Whether the distribution should be recorded in the outgoing mail status.

0	No
1	Yes

Mail reference document name. The document name where a mail item is referred.

Mail reference folder name. The folder name where the referenced document is filed.

Merge type. The value identifies the way the documents are merged. The values are:

1	Query
2	Document
3	File and Blank

Memo slip character set. The graphic character set ID of your memo slip. A graphic character set ID is an identifier used to specify a set of graphic characters in a code page.

Memo slip code page. The graphic code page of your memo slip. A graphic code page is a particular assignment of hexadecimal identifiers to graphic characters.

Multiple line report. The value Y (yes) means you want to create a multiple line report. A multiple line report merges Data Field instructions to create a report where each record of data produces several lines of output.

The value N (no) means you do not want to create a multiple line report.

Number of addressees. The number of user ID (DEN) and user address (DGN) pairs considered primary or addressee recipients contained in DOCI0900 format. The maximum is 32 767.

Number of blind copy list recipients. The number of user ID (DEN) and user address (DGN) pairs considered blind copy list recipients contained in DOCI0900 format. The maximum is 32 767.

Number of copies to print. The quantity of copies you want printed. The valid values are from 1 through 99.

Number of copy list recipients. The number of user ID (DEN) and user address (DGN) pairs that are considered copy list recipients contained in DOCI0900 format. The maximum is 32 767.

Number of labels. The value that identifies the amount or quantity of labels on a page. The valid values are from 1 through 99.

Number of rows per sheet. If you selected Y (yes) for the *Sheet feed labels* prompt, this value from 1 through 99, is the number of rows of labels that you want printed on a page.

Offset to recipient list. The byte offset from the beginning of the structure (base 0) to the start of the variable section containing the recipient list.

Output data on exit. Whether the output data is placed to an output file. The values are:

1	Yes
2	No

Output device. A device in a system by which data can be received from the system. The values are:

1	Printer
2	Display
3	File

Output file name. This is the name of the spooled file that is created. The values are *DOC and *FILE.

Output queue library name. The name of the library being used as the current library during the processing of this command. The value is *LIBL.

Output queue name. A list of output files to be printed or displayed. The output queue is used for spooled files. The values are *DEV, *FILE, and *WRKSTN.

Place on job queue. A document can be merged on the job queue.

Use the value Y (yes) if you want the document merged on the job queue. You can then continue working on your display while the document is being merged on the job queue.

Use the value N (no) if you want the job to merge interactively.

Print change symbols. The value Y (yes) prints revision symbols in the left margin of your document. Revision or change symbols are used to indicate lines that have been revised since the last time the document was changed. The revision symbol character is specified on the Change Editing Options display.

Use N (no) if you do not want to print the revision symbols in the left margin of your document.

Print error log. The value Y (yes) prints the error log page at the end of your document. The log contains any errors that were found while the document was being prepared for printing or errors found when a document was sent from another system. The log can also contain informational messages. If your document does not contain print errors, no log will be printed.

Print from pages 1 through 7. A number from 0.01 through 9999.99 to specify the page on which you want printing to start. Other valid values are *FIRST, *LAST, or *STRPAGE. If you use *FIRST, printing is started on the first page of the document. If you use *LAST, printing is started on the last page of the document. If you use *STRPAGE, the *To page* and *From page* values are the same and only one page is printed. The *From page* and *To page* prompts tell the printer to print specific pages from your document. The page values specified are the pages from the printed document.

Print line numbers. The value Y (yes) prints line numbers in your document. The line numbers begin with 1 on the first page of your document. Line numbers are not printed in headers or footers.

The value N (no) means you do not want to print line numbers in your document.

Print on both sides. The value indicates if you want your document printed on one side or both sides of a page. The values are:

1	No
2	Yes
3	Tumble

Print quality. The value indicates the quality of printing you want to use. The values are:

1 Letter

Letter-quality type is better print but causes wear on the printer ribbon (if your printer uses a ribbon) because the ribbon is struck harder.

2 Text

Text-quality type is not as good as letter-quality type, but is better than draft-quality type.

3 Draft

Draft-quality type saves wear on the printer ribbon (if your printer uses a ribbon) because the ribbon is not struck as hard as it would be if you were using letter-quality type or text-quality type.

Print separator page. A value that determines if sheets will separate the jobs in the printer. The value Y (yes) prints a separator page that includes such things as the document name, folder, document description, subject, reference, and authors. This is useful for separating your document from other documents.

Use N (no) if you do not want to print a separator page.

Print to pages 1 through 7. A number from 0.01 to 9999.99 is the page on which you want printing to stop. Other valid values are *FIRST and *LAST. If you use *FIRST, printing is ended on the first page of the document. If you use *LAST, printing is ended on the last page of the document. The *From page* and *To page* prompts tell the printer to print specific pages from your document. The page values specified are the pages from the printed document.

Printer device name. A device that writes output data from a system on paper or other media.

Printer file library name. The library name of the printer file. Possible values are:

name	The name of the library in which the printer file is stored.
**LIBL*	If you specify *LIBL, your library list is searched first for the printer file.

Printer file name. The name of the file where you want the printed labels to be received and stored. A file is a group of records that are related and treated as a unit. If the file does not exist, a new file will be created for you. If the file already exists, the document will be added to the end of the file.

Query library name. The name of the library that contains the query.

Query name. The name of the query that contains data to be merged.

Reference. The reference of the note or document being passed. This will be blank if the user did not enter a reference.

Reference character set. The graphic character-set ID of the reference. A graphic character-set is an identifier used to specify a set of graphic characters in a code page.

Reference code page. The code page ID of the reference. The code page is a particular assignment of hexadecimal identifiers to graphic characters.

Renumber system page numbers. The value Y (yes) renumbers the system page numbers.

The value N (no) keeps the same system page numbers.

Replace document. Whether the document replaces the existing document (yes) or is added to the existing documents (no).

Replace form. Whether the save form should be replaced if it already exists.

0	No
1	Yes

Replace or add. Whether the data is added or replaced in the file member.

A	Add
R	Replace

Reply requested. Whether the originator has requested a reply.

0	No
1	Yes

Reply requested date. The format is CYYMMDD, where C indicates the century digits. For example, 0 indicates years of the format 19xx, and 1 indicates years of the format 20xx.

Reply requested time. The format is HHMMSS.

Reserved. An ignored field.

Resolve instructions. The value Y (yes) processes the instructions that you have placed in your document.

Resolved output document. The saved document is a resolved document (saved in printed form), that is, page endings are adjusted, line endings are adjusted, headers and footers are put in, and instructions are processed (optional). If you wish to print this document again, it takes less time to print because this document is already in printed form. (A resolved document is a document with the text instructions processed.) This document can be from the text editor or another system.

Resolved output folder. The name of the folder that will contain the document you specified in the *Document name* prompt. If the folder does not exist, a message is shown before it is created.

Save folder. The folder name where the save form document is saved.

Save form. The name of the document to save the results of the fill form process.

Save resolved output. The value Y (yes) means the document you are printing is saved in the document specified in the document prompt.

The value N (no) means you do not want the document you are printing saved in another document.

Send completion message. The value Y (yes) means you are putting your print job on the job queue and you want a message sent to your display station when the job has completed.

The value N (no) means you are putting your print job on the job queue and you do not want a message sent to your display station when the job has completed.

Sensitivity. An indicator of the sensitivity of the content of the note. Values are as follows:

1	None
2	Personal
3	Private
4	Confidential

Shell folder. The folder containing the shell form.

Shell form. The shell document for fill form.

Sheet-feed labels. A value that defines that a device is attached to a printer to automatically feed out sheets of labels. The value Y (yes) means you are sheet-feed printing and want more than one row of labels on a page. If you are using sheet-feed paper, there is no other way to print more than one row of labels on a page.

Show exit display. Whether you want the exit display shown. The valid values are Y (yes) or N (no).

Size of format. The complete size of format DOCI0900, including any variable length records.

Subject. The subject of the note or document being passed. This will be blank if the user did not enter a subject.

Subject character set. The graphic character set ID of the subject. A graphic character set is an identifier used to specify a set of graphic characters in a code page.

Subject code page. The code page ID of the subject. The code page is a particular assignment of hexadecimal identifiers to graphic characters.

To document name. The name of the document to be merged with.

To folder name. The name of the folder that will contain the document being created.

Type of check. The value indicates if you want to check the spelling or the grade level.

0	Spell

1	Grade

User address. The recipient address or DGN.

User ID. The recipient name or DEN.

User's memo slip text. User's text typed on the *Attach Memo Slip* display.

Width of labels. The width of a label is the number of characters from the left edge of the first label to the left edge of the next label, including the blank spaces between the labels. If the width you specify is larger than the margins for your document, the margins are used as the width. Valid values are 2 through 198.

Note: For the printer device field the special values *SYSVAL, *USRPRF, and *WRKSTN are replaced with the appropriate printer name, therefore the exit program does not use the special values.

Error Messages

Message ID	Error Message Text
OFC169A	User exit programs canceled by program &1 in library &2.
OFC169B	Error with administration user exit programs.
OFC1691	Error occurred using Office application enabler.
OFC1695	Error occurred when calling program &1 in library &2.
OFC1696	Return code &1 from program &2 in library &3 not valid.
OFC1698	Error calling user exit program &1 in library &2.
OFC1699	Error on retrieval of user exit program.
OFC2096	The user application send processing has returned &1.

Exit program introduced: V2R2

User Application Administration Exit Program

Required Parameter Group:

1	Exit processing indicator	Output	Char(4)

```
Exit Point Name:  QIBM_QOE_USR_ADM

Exit Point Format Name:  UADM0100
```

The main administration panel has option 50 (User Application Administration) that passes control to the application enabler. One or more of the registered alternate administration programs is then called.

Program Registration

To register user exit programs, see Registration Facility APIs. If you use the callable API, be sure to specify all 20 characters (the 19-character API followed by a blank character).

Required Parameter

Exit processing indicator

OUTPUT; CHAR(4)

This is provided for communication from the user exit program back to the OfficeVision programs. The valid values are listed later.

Any code returned that is not valid will cause an information message to be sent to the job log. This message will identify the invalid return code received, the user exit program name, and the library name that issued the invalid return code. After the information message has been sent to the job log, processing will continue as if a 0000 return code had been returned from the user exit program.

An exception condition returned from the user exit program to the OfficeVision program will post an information message in the job log. This message will attempt to identify the user exit program name and the library name that caused the exception. After the information message has been sent to the job log, processing will continue as if a 0000 return code had been returned from the user exit program.

0000	No additional processing is required for this request. That is, processing is complete for the user exit program.

0002 The user exit program requests cancelation of calling additional user exit programs. This provides the opportunity for a user exit program to support an end user request to stop processing any additional user exit programs and immediately return to the administration main menu. The user exit program should take into consideration that the end user may not understand the impact of the cancel request. Also, consider that the administrator has the opportunity to change the user exit program numbers and may cause a change in the processing sequence of multiple user exit programs.

A message will be put in the job log identifying the user exit program and library that requested this cancelation.

Processing of multiple user exit programs

The user may have multiple user exit programs registered for this exit point. Each exit program will have an *exit program number* defined when it is registered.

- The lowest numbered registered exit program will be called first.
- The second lowest numbered registered exit program will be called.
- This process will continue until all of the registered user exit programs for this exit point have been called.

Exit Program Introduced: V3R1

AnyMail/400 Mail Server Framework APIs

The mail server framework defines the format of how the data is passed rather than the format of the data itself. This allows it to work independently of the format of the data. The format of the data is defined by assigning a type to the information. The different types supported include:

- Address type
- Envelope type
- Attachment reference type
- Message type

For additional information, see Using AnyMail/400 Mail Server Framework APIs

The mail server framework APIs consist of the program calls shown in the following list:

- Add Mail Server Framework Configuration (QzmfAddMailCfg) generates a configuration for the mail server framework used in processing messages.
- Change Mail Message (QzmfChgMailMsg) changes information about a mail message that was previously created using the Create Mail Message (QzmfCrtMailMsg) API.
- Complete Creation Sequence (QzmfCrtCmpMailMsg) removes a previously created, reserved mail message identifier from the mail server framework's list of reserved identifiers and acknowledges that the message was created.
- Create Mail Message (QzmfCrtMailMsg) creates an electronic mail message.
- List Mail Server Framework Configuration (QzmfLstMailCfg) generates a list of type configurations.
- Query Mail Message Identifier (QzmfQryMailMsgId) queries the status of a message associated with a mail message identifier.
- Remove Mail Server Framework Configuration (QzmfRmvMailCfg) removes a configuration from the mail server framework.
- Remove Reserved Mail Message Identifier (QzmfRsvMailMsgID) removes an identifier for an electronic mail message that has been reserved using the Reserve Mail Message Identifier (QzmfRsvMailMsgId) API, but has not been used to create a message.
- Reserve Mail Message Identifier (QzmfRsvMailMsgId) reserves an identifier for an electronic mail message.
- Retrieve Mail Message (QzmfRtvMailMsg) retrieves information about an electronic mail message and returns it in the receiver variables provided by the caller.

The user can write exit programs that are called by the system for the following mail enablement exit programs:

- Snap-In Call exit program provides information about an electronic mail message to defined snap-in programs.
- Track Mail Message Changes exit program allows exit programs to read all the parts of a message to keep track of the message.
- Validate Data Field exit program allows exit programs to validate the data for a message based on the type selected.

Using AnyMail/400 Mail Server Framework APIs

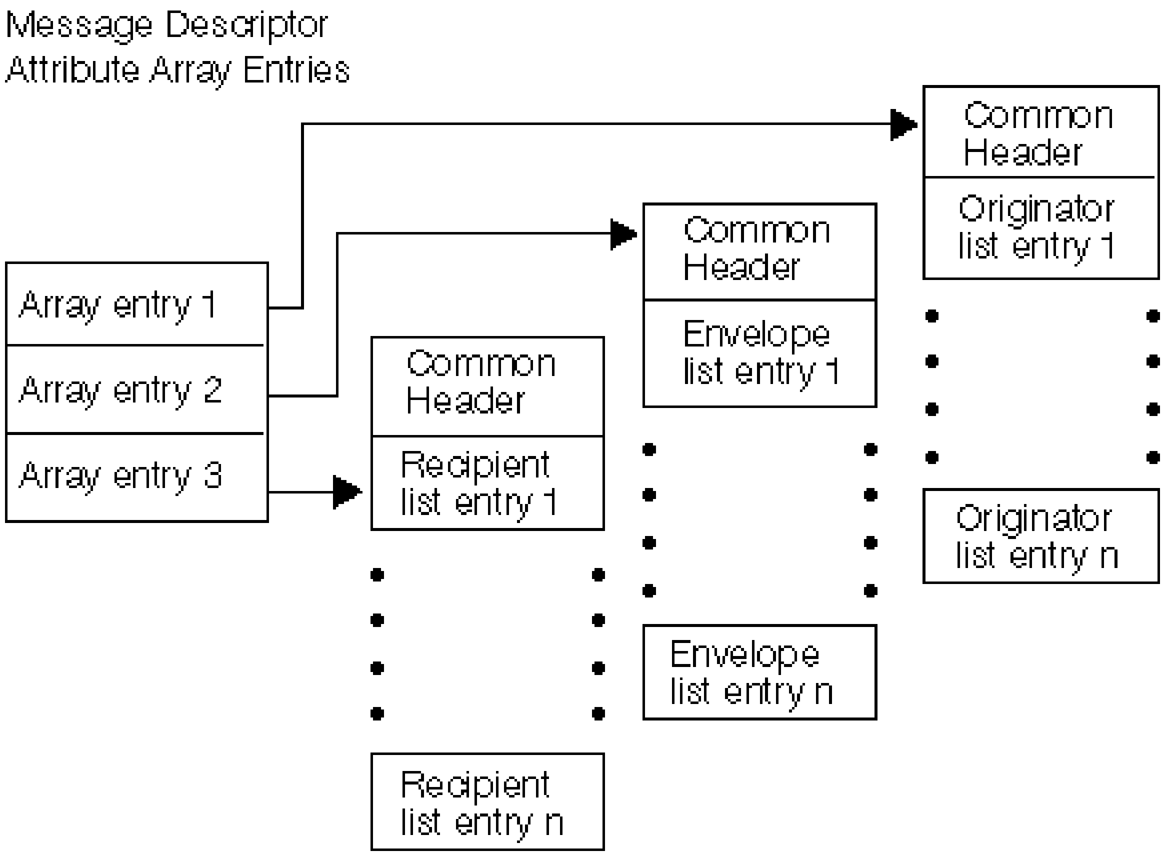
The mail server framework can be configured for up to 128 types in each type group. On each iSeries system, type configuration defines the list of the types that the system can process. The data that is passed to the Create Mail Message API must be in a type that is supported by the system.

A message type is associated with each recipient of the message and is used to tailor the processing of the message. Message types are usually assigned when address resolution takes place.

A message is the sum of all of the data in the message descriptors. These message descriptors are passed to the Create Mail Message (QzmfCrtMailMsg) API using the array of message descriptor attributes.

Figure 1 is an example of a message descriptor that has three descriptor lists being passed: 1) recipient list, 2) envelope list, and 3) originator list. The attachment list, original recipient list, reply-to address list, report-on address list, and report-to address list are optional descriptor lists.

Figure 1. Example of a Message Descriptor for the Create Mail Message (QzmfCrtMailMsg) or Change Mail Message (QzmfChgMailMsg) API



If no errors are encountered during the creation of the mail message, the mail message identifier field contains the unique message identifier associated with the message. If a reserved mail message identifier had been passed, then the reserved mail message identifier is put in the mail message identifier parameter. If any errors are encountered during the creation process, an error message is generated and the mail message identifier returned is set to zeros.

All messages referred to in this section are issued synchronously.

All displacements are base 0 unless otherwise specified.

The AnyMail/400 Mail Server Framework Developer Guide, GG24-4449, focuses on the technical details needed to understand how the mail server framework works. It provides information about how to write programs that create MSF messages or act on MSF messages as MSF exit point programs.

Add Mail Server Framework Configuration (QzmfAddMailCfg) API

Required Parameter Group:

1	Type configuration	Input	Char(*)
2	Format name	Input	Char(8)
3	Error code	I/O	CHAR(*)

Service Program: QZMFASRV

Threadsafe: No

The Add Mail Server Framework Configuration (QzmfAddMailCfg) API generates a configuration for the mail server framework used in validating the information in a message.

Required Parameter Group

Type configuration

INPUT; CHAR(*)

The structure containing the information used for the mail server framework type being configured.

Format name

INPUT; CHAR(8)

The format of the type configuration parameter being used as input. This must be set to ADDC0100.

Error code

I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter.

ADDC0100 Format

The following table shows the layout of the parameter structure for ADDC0100. For a detailed description of each field, see Field Descriptions.

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	BINARY(4)	Length of structure
4	4	CHAR(2)	Type group
6	6	CHAR(4)	Type value

10	A	CHAR(8)	Type name
18	12	CHAR(2)	Reserved (must be blank)
20	14	BINARY(4)	Type text coded character set identifier (CCSID)
24	18	CHAR(100)	Type text

Field Descriptions

Length of structure. The length in bytes of the add mail configuration parameter list.

Reserved. This is a reserved field that must be set to blank.

Type group. The type group being configured. This is a 2-character field representing the type group being configured. This field must be set to one of the following values:

01	Address type group
02	Message type group
03	Envelope type group
04	Attachment type group

There is a maximum of 128 different type value entries in each type group. If an attempt is made to configure more than 128 type values in a type group, error message CPFAFB2 is generated.

Type name. The name of the type being configured. This is an 8-character mnemonic representation of the type. This must be a unique type name. Allowed character values are A through Z and 0 through 9. This field must not be set to blanks. If this field is blank, error message CPFAFB0 is generated. This value may be used for messages when referring to the type.

Type text. The description of the type being configured. If this field is not used, it is set to blanks.

Type text coded character set identifier (CCSID). The coded character set identifier (CCSID) of the type text field. Valid CCSID values for this field are 1 through 65533 and 65535. If this field is set to zero, then the job CCSID is used.

Type value. The value of the type being configured. This is a unique 4-character representation of the type being configured. Allowed character values are A through Z and 0 through 9. It is used in the registration of exit point programs for the mail server framework exit points. The type value is used in the program data fields to indicate whether a program should be called when the mail server framework is processing a specific type of message. The following type values are reserved:

0xxx-1xxx	Reserved for use by IBM.
9998	Reserved for use as the nondelivery message type.
9999	Reserved for use in specifying that a program is configured to process all types.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPF3C36 E	Number of parameters, &1, entered for this API was not valid.
CPF3C90 E	Literal value cannot be changed.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.
CPFAFB0 E	Mail server framework configuration API error occurred.
CPFAFB2 E	Maximum types for type group &1 already exists.

API Introduced: V3R1

Change Mail Message (QzmfChgMailMsg) API

Required Parameter Group:

1	Mail message identifier	Input	Char(32)
2	Message descriptor attributes	Input	Array of Char(*)
3	Number of message descriptor attributes	Input	Binary(4)
4	Format name	Input	Char(8)
5	Error code	I/O	Char(*)

Service Program: QZMFASRV

Threadsafe: No

The Change Mail Message (QzmfChgMailMsg) API changes information about a mail message that was previously created using the Create Mail Message (QzmfCrtMailMsg) API. The Change Mail Message (QzmfChgMailMsg) API can be used to change one or more list items for each of the message descriptors on a single call. This means that one call can result in changes, for example, to the envelope list, the recipient list, and the attachment reference list. The Change Mail Message (QzmfChgMailMsg) API first checks if the caller has the proper authority to request the changes. If any rule is violated, then the entire change fails. If the requested changes are acceptable, the changes that are specified are made to the list. Each list item does not retain its same unique list identifier after the change is completed.

When processing recipient entries using this API, the framework can be instructed to put a recipient entry back so that it can be processed by other snap-in programs. This causes the recipient entry to be processed again. A recipient is marked to be processed again by setting the recipient status to -1.

Note: The Change Mail Message (QzmfChgMailMsg) API can only be called from a Snap-In exit point.

Restriction: Originator, report-on, and report-to entries can be added but not changed. The unique identifier fields for these list entries must be -1.

Required Parameter Group

Mail message identifier

INPUT; CHAR(32)

The identifier of the mail message to be changed. The mail message identifier is composed of characters A through Z and 0 through 9 only.

Message descriptor attributes

INPUT; ARRAY OF CHAR(*)

This array contains pointers to the message descriptor attributes. It contains lengths of these message descriptor attributes, along with the format name of the data that is provided (see Figure 1). The format variable within this parameter list indicates to what type of list the pointer refers. See Change Message

Common Header for a description of the header data associated with each parameter list.

Number of message descriptor attributes

INPUT; BINARY(4)

The number of message descriptor attributes to change. At least one message descriptor attribute must be specified on a call to this API.

Format name

INPUT; CHAR(8)

The format of the message descriptor attributes that is being passed. This field must be set to CHGM0100.

Error code

I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter.

CHGM0100 Format

The following table shows the data that must be provided for each message descriptor attribute.

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	POINTER	Message descriptor pointer
16	10	BINARY(4)	Length of the message descriptor
20	14	CHAR(8)	Message descriptor format name
28	1C	BINARY(4)	Reserved (must be zero)

Message Descriptor

Each entry of the message descriptor attributes consists of:

- A change message common header section
- Message descriptor format. The following table shows the format names of the valid message descriptors.

Format name	Description
ORCL0100	Original recipient entry
ORGL0100	Originator entry
ENVL0100	Envelope entry
RCPL0100	Recipient entry
ROAL0100	Report-on address entry

RPYL0100	Reply-to address entry
RTAL0100	Report-to address entry
ATTL0100	Attachment reference entry

Change Message Common Header

The following table shows the common header section that is always included at the beginning of each entry.

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	BINARY(4)	Length of this message descriptor
4	4	BINARY(4)	Reserved (must be set to zero)
8	8	CHAR(8)	Message descriptor format name.
16	10	BINARY(4)	Offset of the first entry in the message descriptor
20	14	BINARY(4)	Number of entries in this message descriptor
24	18	BINARY(4)	Reserved (must be set to zero)
*	*	CHAR(*)	Message descriptor data.

The common header is used for all of the different formats that can be supplied on the Change Mail Message (QzmfChgMailMsg) API. The common header must always be included. There must be one or more list items after the common header section.

The list item or items to be changed are identified using the unique list identifier that is assigned by the mail server framework. The unique list identifiers are part of the lists that are passed to the exit programs or are part of the information returned by the Retrieve Mail Message (QzmfRtvMailMsg) API. The unique list identifiers are also passed to snap-in exit programs. For a new addition to the list, the parameter list unique identifier must be set to -1. There is no way to remove an entry using this API. Each list must contain unique identifiers that are in ascending order, with add entries (unique identifier of -1) at the end of the list.

For substitution changes, a parameter list unique identifier must be specified once, along with the data that is to be substituted. For expansion changes, a parameter list unique identifier must be specified multiple times, once for each expansion list item that is to be added. After the API has completed, the unique identifiers that were changed are no longer valid and may not be used in future calls to this API. The retrieve mail message (QzmfRtvMailMsg) API passes lists with entries sorted so that unique identifiers are in ascending order. Snap-in exit programs also receive sorted lists.

Message Descriptor Formats

One of the following message descriptor formats must be used to describe the data that follows the common header. There may be multiple instances of the formats following the common header, depending on the size of the parameter list.

ORCL0100 Format (Original Recipient Entry)

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Length of this entry
4	4	BINARY(4)	Displacement of the original recipient address from the beginning of this ORCL0100 entry
8	8	BINARY(4)	Length of the original recipient address
12	C	CHAR(4)	Address type
16	10	BINARY(4)	Original recipient address coded character set identifier(CCSID)
20	14	BINARY(4)	Distribution type
24	18	BINARY(4)	Reply requested flag
28	1C	BINARY(4)	Unique identifier
32	20	BINARY(4)	Unique identifier of referenced ORCL0100 entry
36	24	BINARY(4)	Reserved (must be set to zero)
*	*	CHAR(*)	Original recipient address

ORGL0100 Format (Originator Entry)

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Length of this entry
4	4	BINARY(4)	Displacement of the origin address from the beginning of this ORGL0100 entry
8	8	BINARY(4)	Length of origin address
12	C	CHAR(4)	Address type
16	10	BINARY(4)	Origin address coded character set identifier (CCSID)
20	14	BINARY(4)	Unique identifier
24	18	BINARY(4)	Unique identifier of referenced ORGL0100 entry
28	1C	BINARY(4)	Reserved (must be set to zero)
*	*	CHAR(*)	Origin address

ENVL0100 Format (Envelope Entry)

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Length of this entry

4	4	BINARY(4)	Displacement of the envelope from the beginning of this ENVL0100 entry
8	8	BINARY(4)	Length of envelope
12	C	CHAR(4)	Envelope type
16	10	BINARY(4)	Unique identifier
20	14	BINARY(4)	Unique identifier of referenced ENVL0100 entry
24	18	BINARY(4)	Reserved (must be set to zero)
*	*	CHAR(*)	Envelope

RCPL0100 Format (Recipient Entry)

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	BINARY(4)	Length of this entry
4	4	BINARY(4)	Displacement of the snap-in provided information (SPIN) from the beginning of this RCPL0100 entry
8	8	BINARY(4)	Length of snap-in provided information (SPIN)
12	C	BINARY(4)	Displacement of the original recipient address from the beginning of this RCPL0100 entry
16	10	BINARY(4)	Length of recipient address
20	14	CHAR(4)	Address type
24	18	BINARY(4)	Recipient address coded character set identifier (CCSID)
28	1C	BINARY(4)	Reason code
32	20	BINARY(4)	Diagnostic code
36	24	CHAR(4)	Message type
40	28	BINARY(4)	Status
44	2C	BINARY(4)	Reserved
48	30	BINARY(4)	Unique identifier
52	34	BINARY(4)	Reserved (must be set to zero)
*	*	CHAR(*)	Recipient address
*	*	CHAR(*)	Snap-in provided information (SPIN)

ROAL0100 Format (Report-on Address Entry)

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	BINARY(4)	Length of this entry

4	4	BINARY(4)	Displacement of the snap-in provided information (SPIN) from the beginning of this ROAL0100 entry
8	8	BINARY(4)	Length of snap-in provided information (SPIN)
12	C	BINARY(4)	Displacement of the report-on address from the beginning of this ROAL0100 entry
16	10	BINARY(4)	Length of address
20	14	CHAR(4)	Address type
24	18	BINARY(4)	Report-on address coded character set identifier (CCSID)
28	1C	BINARY(4)	Reason code
32	20	BINARY(4)	Diagnostic code
36	24	BINARY(4)	Reserved
40	28	BINARY(4)	Unique identifier
44	2C	BINARY(4)	Unique identifier of referenced ROAL0100 entry
*	*	CHAR(*)	Report-on address
*	*	CHAR(*)	Snap-in provided information (SPIN)

RPYL0100 Format (Reply-to Address Entry)

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	BINARY(4)	Length of this entry
4	4	BINARY(4)	Displacement of the reply-to address from the beginning of this RPYL0100 entry
8	8	BINARY(4)	Length of reply-to address
12	C	CHAR(4)	Address type
16	10	BINARY(4)	Reply-to address coded character set identifier (CCSID)
20	14	BINARY(4)	Unique identifier
24	18	BINARY(4)	Unique identifier of referenced RPYL0100 entry
28	1C	BINARY(4)	Reserved (must be set to zero)
*	*	CHAR(*)	Reply-to address

RTAL0100 Format (Report-to Address Entry)

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	BINARY(4)	Length of this entry

4	4	BINARY(4)	Displacement of the report-to address from the beginning of this RTAL0100 entry
8	8	BINARY(4)	Length of address
12	C	CHAR(4)	Address type
16	10	BINARY(4)	Report-to address coded character set identifier (CCSID)
20	14	BINARY(4)	Unique identifier
24	18	BINARY(4)	Unique identifier of referenced RTAL0100 entry
28	1C	BINARY(4)	Reserved (must be set to zero)
*	*	CHAR(*)	Report-to address

ATTL0100 Format (Attachment Reference Entry)

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	BINARY(4)	Length of this entry
4	4	BINARY(4)	Displacement of the attachment reference from the beginning of this ATTL0100 entry
8	8	BINARY(4)	Length of attachment reference
12	C	CHAR(4)	Attachment reference type
16	10	BINARY(4)	Unique identifier
20	14	BINARY(4)	Unique identifier of referenced ATTL0100 entry
24	18	BINARY(4)	Reserved (must be set to zero)
*	*	CHAR(*)	Attachment reference

Field Descriptions

Address type. The type of address that is contained in the entry.

Attachment reference. A reference to an attachment that is associated with the message.

Attachment reference type. The type of attachment that is contained in the entry.

Diagnostic code. A code that indicates the cause of a problem which caused the delivery of a message to this address to fail. The diagnostic code is assumed to contain the X.400 nondelivery diagnostic code.

Displacement of the attachment reference from the beginning of this ATTL0100 entry. The displacement from the beginning of this entry to the attachment reference in this entry.

Displacement of the envelope from the beginning of this ENVL0100 entry. The displacement from the beginning of this entry to the envelope in this entry.

Displacement of the original recipient address from the beginning of this ORCL0100 entry. The displacement from the beginning of this entry to the original recipient address in this entry.

Displacement of the origin address from the beginning of this ORGL0100 entry. The displacement from the beginning of this entry to the origin address in this entry.

Displacement of the recipient address from the beginning of this RCPL0100 entry. The displacement from the beginning of this entry to the recipient address in this entry.

Displacement of the report-on address from the beginning of this ROAL0100 entry. The displacement from the beginning of this entry to the report-on address in this entry.

Displacement of the reply-to address from the beginning of this RPYL0100 entry. The displacement from the beginning of this entry to the reply-to address in this entry.

Displacement of the report-to address from the beginning of this RTAL0100 entry. The displacement from the beginning of this entry to the report-to address in this entry.

Distribution type. The type of distribution associated with each recipient entry. The possible values are:

0	Normal message recipient
1	CC (carbon copy) recipient
2	BCC (blind carbon copy) recipient

Envelope. A string of data representing information about the message, aside from the attachments and its recipients.

Envelope type. The type of envelope that is contained in the entry.

Format name. The content and format of the information provided for each message parameter. The possible values are:

ORCL0100	Original recipient entry
ORGL0100	Originator entry
ENVL0100	Envelope entry
RCPL0100	Recipient entry
ROAL0100	Report-on address entry
RPYL0100	Reply-to address entry
RTAL0100	Report-to address entry
ATTL0100	Attachment reference entry

Length of address. The length of address that is contained in the entry. The maximum length of an address is 1024 bytes.

Length of attachment reference. The length in bytes of the attachment reference that is contained in the entry.

Length of envelope. The length of envelope that is contained in the entry.

Length of message descriptor. The length in bytes of the message descriptor that is being pointed to by the pointer to the message descriptor. The maximum length of a message descriptor is 16 million bytes.

Length of origin address. The length of the origin address for this entry. The maximum length of an origin address is 1024 bytes.

Length of original recipient address. The length of the original recipient address for this entry. The maximum length of an original recipient address is 1024 bytes.

Length of recipient address. The length in bytes of the recipient address. The maximum length of a recipient address is 1024 bytes.

Length of reply-to address. The length in bytes of the reply-to address. The maximum length of a reply-to address is 1024 bytes.

Length of snap-in provided information (SPIN). The length in bytes of the snap-in provided information (SPIN). The maximum length of the SPIN is 256 bytes.

Length of this entry. The length in bytes of this entry. This is used to get to the next entry.

Length of this message descriptor. The length in bytes of this message descriptor. The maximum length of a message descriptor is 16 million bytes.

Message descriptor data. One or more parameter list formats that follow the common header. Message descriptors are made up of a common header and a list of entries. The format of each entry in the list is defined by the format name associated with the message descriptor, which is located in the common header. The number of entries in the list is also defined in the common header.

Message type. The type of message that is associated with the entry.

Number of bytes available for this message descriptor. The number of bytes available in the space where the mail server framework puts the information being retrieved.

Offset of the first entry in the message descriptor. The offset from the beginning of this message descriptor to the first entry in the list of entries.

Origin address. A string that represents the address associated with the originator of the message. The contents and format of the string are not defined by the mail server framework. It is assumed that the address type associated with the origin address defines the contents of the origin address field.

Origin address coded character set identifier (CCSID). The CCSID provided for the origin address. Valid values for the CCSID are 1 through 65533 and 65535.

Original recipient address. A string that represents the address associated with the original recipient of the message. The contents and format of the string are not defined by the mail server framework. It is assumed that the address type associated with the original recipient address defines the contents of the original recipient address field.

Original recipient address coded character set identifier (CCSID). The CCSID provided for the original recipient address. Valid values for the CCSID are 1 through 65533 and 65535.

Reason code. A code that identifies reasons associated with the message delivery to this address. In the case of a nondelivery entry, this field would contain the reason the delivery of this message to this recipient failed. The reason code is assumed to contain the X.400 nondelivery reason code.

Recipient address. A string that represents the address associated with a recipient of the message. The contents and format of the string are not defined by the mail server framework. It is assumed that the address type associated with the recipient address defines the contents of the recipient address field.

Recipient address coded character set identifier (CCSID). The CCSID provided for the recipient address. Valid values for the CCSID are 1 through 65533 and 65535.

Reply requested flag. Whether this original recipient should reply to the message. The possible values are as follows:

0	A reply is not requested from this original recipient
1	A reply is requested from this recipientz

Reply-to address. A string that represents the address to be replied to. The contents and format of the string are not defined by the mail server framework. It is assumed that the address type associated with the reply-to address defines the contents of the reply-to address field.

Reply-to address coded character set identifier (CCSID). The CCSID provided for the reply-to address. Valid values for the CCSID are 1 through 65533 and 65535.

Report-on address. A string which represents the address to be reported on. The contents and format of the string are not defined by the mail server framework. It is assumed that the address type associated with the report-on address defines the contents of the report-on address field.

Report-on address coded character set identifier (CCSID). The CCSID provided for the report-on address. Valid values for the CCSID are 1 through 65533 and 65535.

Report-to address. A string which represents the address to be reported to. The contents and format of the string are not defined by the mail server framework. It is assumed that the address type associated with the report-to address defines the contents of the report-to address field.

Report-to address coded character set identifier (CCSID). The CCSID provided for the report-to address. Valid values for the CCSID are 1 through 65533 and 65535.

Reserved. All reserved fields must be set to zero.

Snap-in provided information (SPIN). An area where snap-ins can store information that other snap-ins may use. SPIN provides a place where information relating to a specific recipient can be stored and used by snap-ins in the same or different exit points. This is completely user-defined and user-interpreted data.

Status. The status associated with each recipient entry. The possible values are:

1	Forwarded (remote)
2	Ignore
3	Local
4	Nondeliverable
5	Security violation

Unique identifier. A unique identifier that differentiates each item within a particular list. Identifiers are generated for each list item when the Create Mail Message (QzmfCrtMailMsg) API has successfully completed. These unique identifiers are temporary and may change as the mail service processes a message. List entries are placed in message descriptors such that unique identifiers are in ascending order.

Unique identifier of parent entry. The unique identifier associated with the parent entry of this entry. A parent entry is an entry that had been replaced by a single or multiple entries.

Unique identifier of referenced entry. The unique identifier of another ORCL0100, ORGL0100, ENVL0100, RTAL0100, RPYL0100, ROAL0100, or ATTL0100 entry that this entry refers to. This field can be used to create entry cross references as new entries are added.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3C90 E	Literal value cannot be changed.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.
CPFAF80 E	Syntax error on call to MSF API. Reason code &1.
CPFAF81 E	Parameter value error on call to MSF API. Reason code &1.
CPFAF82 E	Error occurred during running of MSF API.
CPFAF83 E	Parameter error on call to MSF API. Reason code &1.
CPFAF84 E	MSF API failed. MSF message identifier not found.
CPFAF85 E	MSF API failed. Request not allowed.
CPFAF86 E	MSF API failed. Parameter values not valid for exit point.
CPFAF87 E	MSF validate data field exit program error occurred.
CPFAF88 E	Error occurred on call to MSF exit point program.
CPFAF8A E	MSF API failed. List cannot be expanded or replaced.

API Introduced: V3R1

Top | Office APIs | APIs by category

Complete Creation Sequence (QzmfCrtCmpMailMsg) API

Required Parameter Group:

1	Reserved mail message identifier	Input	Char(32)
2	Format name	Input	Char(8)
3	Error code	I/O	Char(*)

Service Program: QZMFASRV

Threadsafe: No

The Complete Creation Sequence (QzmfCrtCmpMailMsg) API is used to remove a reserved mail message identifier from the mail server framework's list of reserved identifiers. This API marks the completion of a mail message creation sequence. This API is to be used after the message associated with the reserved mail message identifier has been created. This API ensures data integrity within the mail framework.

Required Parameter Group

Reserved mail message identifier

INPUT; CHAR(32)

The unique message identifier being completed. The message identifier is composed of characters A through Z and 0 through 9 only.

Format name

INPUT; CHAR(8)

The content and format of the information being passed to the API. This field must be set to CCMP0100.

Error code

I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3C90 E	Literal value cannot be changed.

CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.
CPFAF82 E	Error occurred during running of MSF API.
CPFAF83 E	Parameter error on call to MSF API. Reason code &1.
CPFAF8B E	Reserved mail message identifier is not valid.
CPFAF8C E	Reserved mail message identifier not used.

API Introduced: V3R1

Create Mail Message (QzmfCrtMailMsg) API

Required Parameter Group:

1	Mail message identifier	Output	Char(32)
2	Reserved mail message identifier	Input	Char(32)
3	Creation message type	Input	Char(4)
4	Message descriptor attributes	Input	Array of Char(*)
5	Number of message descriptor attributes	Input	Binary(4)
6	Format name	Input	Char(8)
7	Error code	I/O	Char(*)

Service Program: QZMFASRV

Threadsafe: No

The Create Mail Message (QzmfCrtMailMsg) API creates a single electronic mail message.

The input to this API is a parameter list which includes an array of message descriptor attributes. Message descriptor attributes define the basic characteristics of the message descriptor such as the format name associated with it, a pointer to it, and its length. Each message descriptor has a common header, which defines such things as the number of entries in the message descriptor and where the first entry is located. Following the common header is a list of individual entries, which contain information that define the message. The format name defines the format of the individual entries in the message descriptor data. The following message descriptor formats are used.

Original recipient entry. An entry that contains the addresses of the original recipients of the message.

Originator entry. An entry that contains the addresses of the originators of the message.

Envelope entry. An entry that defines the format that is used to pass envelopes. An envelope is a string of data representing information about the message, aside from the attachments and its recipients. This may include the text of the message.

Recipient entry. An entry that contains information about the recipient of the message. This includes the address of the recipient. Recipient addresses are reported on when the message cannot be delivered to the recipient.

Reply-to address entry. An entry that contains the address to be replied to.

Report-on address entry. An entry that defines the address to be reported on.

Report-to address entry. An entry that defines an address to be reported to in case errors are encountered delivering messages to a recipient.

Attachment reference entry. A reference to an attachment that is associated with the message. Attachments may include the text of the message.

Required Parameter Group

Mail message identifier

OUTPUT; CHAR(32)

The variable containing a unique message identifier associated with this message. The mail message identifier is composed of characters A through Z and 0 through 9 only. On completion of this API, if a reserved mail message identifier field is not set to blanks, this variable contains the reserved mail message identifier. Otherwise, this field contains the identifier associated with the message created by the API. If an error occurs during processing, the identifier is set to zeros and a notification of the error is made in the error code parameter. See Reserve Mail Message Identifier (QzmfRsvMailMsgId) API for information on reserving mail message identifiers.

Reserved mail message identifier

INPUT; CHAR(32)

The variable containing the reserved mail message identifier associated with this message. If the reserved mail message identifier field is set to blanks, then a unique mail message identifier associated with this message is returned in the mail message identifier field after the API has completed running and has created a message. If the reserved mail message identifier field is not set to blanks, then it is assumed that the field contains a reserved mail message identifier. The reserved mail message identifier is returned in the mail message identifier variable when the API has completed successfully. If the value of the reserved mail message identifier field is not a reserved mail message identifier, then error message CPFAF8B is generated. See Reserve Mail Message Identifier (QzmfRsvMailMsgId) API for information on reserving mail message identifiers.

Creation message type

INPUT; CHAR(4)

The message type that the originator intended to create. This is used to indicate the message type associated with the mail server framework message.

Message descriptor attributes

INPUT; CHAR(*)

An array containin message descriptor attributes entries (see Figure 1). A message descriptor attributes entry is required for each of the following message descriptors:

- Originator entry
- Envelope entry
- Recipient entry

The following table shows the data that must be provided for each message descriptor attributes entry. The other message descriptors are optional.

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	POINTER	Message descriptor pointer
16	10	BINARY(4)	Length of the message descriptor
20	14	CHAR(8)	Message descriptor format name

28	1C	BINARY(4)	Reserved (must be zero)

Number of message descriptor attributes

INPUT; BINARY(4)

The number of entries in the array of message descriptor attributes. A minimum of three message descriptor attributes entries is required (originator entry, envelope entry, and recipient entry).

Format name

INPUT; CHAR(8)

The variable containing the format name of the parameter list passed to the API. This field must be set to CRTM0100.

Error code

I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter.

Message Descriptors

A message descriptor consists of:

- A create message common header section
- The actual message descriptor format. The following table shows valid message descriptor formats.

Format name	Description
ORCL0100	Original recipient entry
ORGL0100	Originator entry
ENVL0100	Envelope entry
RCPL0100	Recipient entry
ROAL0100	Report-on address entry
RPYL0100	Reply-to address entry
RTAL0100	Report-to address entry
ATTL0100	Attachment reference entry

Create Message Common Header

The following table shows the common header section that is always included at the beginning of a message descriptor.

Offset			
Dec	Hex	Type	Field
0	0	BINARY(4)	Length of this message descriptor

4	4	BINARY(4)	Reserved (must be set to zero)
8	8	CHAR(8)	Message descriptor format name
16	10	BINARY(4)	Offset of the first entry in the message descriptor
20	14	BINARY(4)	Number of entries in this message descriptor
24	18	BINARY(4)	Reserved (must be set to zero)
*	*	CHAR(*)	Message descriptor data

The common header is used for all of the different formats that can be supplied on the API. A specific format name can only be supplied in one of the message descriptor common headers when this API is called. There may be zero or more list items after the common header section.

Message Descriptor Formats for Create Mail Message

If there is data after the common header, one of the following message descriptor formats must be used. There may be multiple instances of the formats following the common header, depending on the size of the message descriptor.

ORCL0100 Format (Original Recipient Entry)

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	BINARY(4)	Length of this entry
4	4	BINARY(4)	Displacement of the original recipient address from the beginning of this ORCL0100 entry
8	8	BINARY(4)	Length of the original recipient address
12	C	CHAR(4)	Address type
16	10	BINARY(4)	Original recipient address coded character set identifier(CCSID)
20	14	BINARY(4)	Distribution type
24	18	BINARY(4)	Reply requested flag
28	1C	BINARY(4)	Reserved (must be set to zero)
32	20	BINARY(4)	Reserved (must be set to zero)
36	24	BINARY(4)	Reserved (must be set to zero)
*	*	CHAR(*)	Original recipient address

ORGL0100 Format (Originator Entry)

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	BINARY(4)	Length of this entry

4	4	BINARY(4)	Displacement of the origin address from the beginning of this ORGL0100 entry
8	8	BINARY(4)	Length of origin address
12	C	CHAR(4)	Address type
16	10	BINARY(4)	Origin address coded character set identifier (CCSID)
20	14	BINARY(4)	Reserved (set to zero)
24	18	BINARY(4)	Reserved (set to zero)
28	1C	BINARY(4)	Reserved (set to zero)
*	*	CHAR(*)	Origin address

ENVL0100 Format (Envelope Entry)

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	BINARY(4)	Length of this entry
4	4	BINARY(4)	Displacement of envelope from the beginning of this ENVL0100 entry
8	8	BINARY(4)	Length of envelope
12	C	CHAR(4)	Envelope type
16	10	BINARY(4)	Reserved (set to zero)
20	14	BINARY(4)	Reserved (set to zero)
24	18	BINARY(4)	Reserved (set to zero)
*	*	CHAR(*)	Envelope

RCPL0100 Format (Recipient Entry)

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	BINARY(4)	Length of this entry
4	4	BINARY(4)	Displacement of the snap-in provided information (SPIN) from the beginning of this RCPL0100 entry
8	8	BINARY(4)	Length of snap-in provided information (SPIN)
12	C	BINARY(4)	Displacement of the recipient address from the beginning of this RCPL0100 entry
16	10	BINARY(4)	Length of recipient address
20	14	CHAR(4)	Address type
24	18	BINARY(4)	Recipient address coded character set identifier (CCSID)

28	1C	BINARY(4)	Reason code
32	20	BINARY(4)	Diagnostic code
36	24	CHAR(4)	Message type
40	28	BINARY(4)	Status
44	2C	BINARY(4)	Reserved
48	30	BINARY(4)	Reserved (set to zero)
52	34	BINARY(4)	Reserved (set to zero)
*	*	CHAR(*)	Recipient address
*	*	CHAR(*)	Snap-in provided information (SPIN)

ROAL0100 Format (Report-on Address Entry)

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	BINARY(4)	Length of this entry
4	4	BINARY(4)	Displacement of the snap-in provided information (SPIN) from the beginning of this ROAL0100 entry
8	8	BINARY(4)	Length of snap-in provided information (SPIN)
12	C	BINARY(4)	Displacement of the report-on address from the beginning of this ROAL0100 entry
16	10	BINARY(4)	Length of address
20	14	CHAR(4)	Address type
24	18	BINARY(4)	Report-on address coded character set identifier (CCSID)
28	1C	BINARY(4)	Reason code
32	20	BINARY(4)	Diagnostic code
36	24	BINARY(4)	Reserved (set to zero)
40	28	BINARY(4)	Reserved (set to zero)
44	2C	BINARY(4)	Reserved (set to zero)
*	*	CHAR(*)	Report-on address
*	*	CHAR(*)	Snap-in provided information (SPIN)

RPYL0100 Format (Reply-to Address Entry)

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	BINARY(4)	Length of this entry

4	4	BINARY(4)	Displacement of the reply-to address from the beginning of this RPYL0100 entry
8	8	BINARY(4)	Length of reply-to address
12	C	CHAR(4)	Address type
16	10	BINARY(4)	Reply-to address coded character set identifier (CCSID)
20	14	BINARY(4)	Reserved (set to zero)
24	18	BINARY(4)	Reserved (set to zero)
28	1C	BINARY(4)	Reserved (set to zero)
*	*	CHAR(*)	Reply-to address

RTAL0100 Format (Report-to Address Entry)

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	BINARY(4)	Length of this entry
4	4	BINARY(4)	Displacement of the report-to address from the beginning of this RTAL0100 entry
8	8	BINARY(4)	Length of address
12	C	CHAR(4)	Address type
16	10	BINARY(4)	Report-to address coded character set identifier (CCSID)
20	14	BINARY(4)	Reserved (set to zero)
24	18	BINARY(4)	Reserved (set to zero)
28	1C	BINARY(4)	Reserved (set to zero)
*	*	CHAR(*)	Report-to address

ATTL0100 Format (Attachment Reference Entry)

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	BINARY(4)	Length of this entry
4	4	BINARY(4)	Displacement of the attachment reference from the beginning of this ATTL0100 entry
8	8	BINARY(4)	Length of attachment reference
12	C	CHAR(4)	Attachment reference type
16	10	BINARY(4)	Reserved (set to zero)
20	14	BINARY(4)	Reserved (set to zero)
24	18	BINARY(4)	Reserved (set to zero)

*	*	CHAR(*)	Attachment reference

Field Descriptions

Address type. The type of address that is contained in the entry.

Attachment reference. A reference to an attachment that is associated with the message.

Attachment reference type. The type of attachment that is contained in the entry.

Diagnostic code. A code that indicates the cause of a problem which caused the delivery of a message to this address to fail. The diagnostic code is assumed to contain the X.400 nondelivery diagnostic code.

Displacement of the attachment reference from the beginning of this ATTL0100 entry. The displacement from the beginning of this entry to the attachment reference in this entry.

Displacement of the envelope from the beginning of this ENVL0100 entry. The displacement from the beginning of this entry to the envelope in this entry.

Displacement of the origin address from the beginning of this ORGL0100 entry. The displacement from the beginning of this entry to the origin address in this entry.

Displacement of the original recipient address from the beginning of this ORCL0100 entry. The displacement from the beginning of this entry to the original recipient address in this entry.

Displacement of the recipient address from the beginning of this RCPL0100 entry. The displacement from the beginning of this entry to the recipient address in this entry.

Displacement of the reply-to address from the beginning of this RPYL0100 entry. The displacement from the beginning of this entry to the reply-to address in this entry.

Displacement of the report-to address from the beginning of this RTAL0100 entry. The displacement from the beginning of this entry to the report-to address in this entry.

Displacement of the report-on address from the beginning of this ROAL0100 entry. The displacement from the beginning of this entry to the report-on address in this entry.

Displacement of the snap-in provided information (SPIN) of this RCHL0100. The displacement from the beginning of this entry to the snap-in provided information (SPIN). (There are fields for RCPL0100 and RCHL0100 entries.)

Distribution type. The type of distribution associated with each recipient entry. The possible values are:

0	Normal message recipient
1	CC (carbon copy) recipient
2	BCC (blind carbon copy) recipient

Envelope. A string of data representing information about the message, aside from the attachments and its recipients.

Envelope type. The type of envelope that is contained in the entry.

Length of address. The length of address that is contained in the entry. The maximum length of an address is 1024 bytes.

Length of attachment reference. The length in bytes of the attachment reference that is contained in the entry.

Length of envelope. The length of envelope that is contained in the entry.

Length of origin address. The length of the origin address for this entry. The maximum length of an origin address is 1024 bytes.

Length of the original recipient address. The length of the original recipient address for this entry. The maximum length of an original recipient address is 1024 bytes.

Length of receiver message descriptor. The length in bytes of the message descriptor that is being pointed to by the pointer to the message descriptor. The maximum length of a message descriptor is 16 million bytes.

Length of recipient address. The length in bytes of the recipient address. The maximum length of a recipient address is 1024 bytes.

Length of reply-to address. The length in bytes of the reply-to address. The maximum length of a reply-to address is 1024 bytes.

Length of snap-in provided information (SPIN). The length in bytes of the snap-in provided information (SPIN). The maximum length of the SPIN is 256 bytes.

Length of this entry. The length in bytes of this entry. This is used to get to the next entry.

Length of this message descriptor. The length in bytes of this message descriptor. The maximum length of a message descriptor is 16 million bytes.

Message descriptor data. One or more parameter list formats that follow the common header. Message descriptors are made up of a common header and a list of entries. The format of each entry in the list is defined by the format name associated with the message descriptor, which is located in the common header. The number of entries in the list is also defined in he common header.

Message descriptor format name. The content and format of the information provided for each message parameter. The possible values are:

ORCL0100	Original recipient entry
ORGL0100	Originator entry
ENVL0100	Envelope entry
RCPL0100	Recipient entry
RPYL0100	Reply-to address entry
ROAL0100	Report-on address entry
RTAL0100	Report-to address entry
ATTL0100	Attachment reference entry

Message descriptor pointer. A space pointer that points to a message descriptor. A message descriptor is used to pass information which describes the message.

Message type. The type of message that is associated with the entry.

Number of bytes available for this message descriptor. The number of bytes available in the space where the mail server framework puts the information being retrieved.

Offset of the first entry in the message descriptor. The offset from the beginning of this message descriptor to the first entry in the list of entries.

Origin address. A string that represents the address associated with the originator of the message. The contents and format of the string are not defined by the mail server framework. It is assumed that the address type associated with the origin address defines the contents of the origin address field.

Origin address coded character set identifier (CCSID). The CCSID provided for the origin address. Valid values for the CCSID are 1 through 65533 and 65535.

Original recipient address. A string that represents the address associated with the original recipient of the message. The contents and format of the string are not defined by the mail server framework. It is assumed that the address type associated with the original recipient address defines the contents of the original recipient address field.

Original recipient address coded character set identifier (CCSID). The CCSID provided for the original recipient address. Valid values for the CCSID are 1 through and including 65533, and 65535.

Reason code. A code that identifies reasons associated with the message delivery to this address. In the case of a nondelivery entry, this field would contain the reason the delivery of this message to this recipient failed. The reason code is assumed to contain the X.400 nondelivery reason code.

Recipient address. A string that represents the address associated with a recipient of the message. The contents and format of the string are not defined by the mail server framework. It is assumed that the address type associated with the recipient address defines the contents of the recipient address field.

Recipient address coded character set identifier (CCSID). The CCSID provided for the recipient address. Valid values for the CCSID are 1 through 65533 and 65535.

Reply requested flag. A flag that indicates whether this original recipient should reply to the message. The possible values are as follows:

0	A reply is not requested from this original recipient
1	A reply is requested from this original recipient

Reply-to address. A string that represents the address to be replied to. The contents and format of the string are not defined by the mail server framework. It is assumed that the address type associated with the reply-to address defines the contents of the reply-to address field.

Reply-to address coded character set identifier (CCSID). The CCSID provided for the reply-to address. Valid values for the CCSID are 1 through and including 65533, and 65535.

Report-on address. A string which represents the address to be reported on. The contents and format of the string are not defined by the mail server framework. It is assumed that the address type associated with the report-on address defines the contents of the report-on address field.

Report-on address coded character set identifier (CCSID). The CCSID provided for the report-on address. Valid values for the CCSID are 1 through 65533 and 65535.

Report-to address. A string which represents the address to be reported to. The contents and format of the string are not defined by the mail server framework. It is assumed that the address type associated with the report-to address defines the contents of the report-to address field.

Report-to address coded character set identifier (CCSID). The CCSID provided for the report-to address. Valid values for the CCSID are 1 through 65533 and 65535.

Reserved. All reserved fields must be set to zero.

Snap-in provided information (SPIN). An area where snap-ins can store information that other snap-ins may use. SPIN provides a place where information relating to a specific recipient can be stored and used by snap-ins in the same or different exit points. This is completely user-defined and user-interpreted data.

Status. The status associated with each recipient entry. The possible values are:

1	Forwarded (remote)
2	Ignore
3	Local
4	Nondeliverable
5	Security violation

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3C90 E	Literal value cannot be changed.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.
CPFAF80 E	Syntax error on call to MSF API. Reason code &1.
CPFAF81 E	Parameter value error on call to MSF API. Reason code &1.
CPFAF82 E	Error occurred during running of MSF API.
CPFAF83 E	Parameter error on call to MSF API. Reason code &1.
CPFAF87 E	MSF validate data field exit program error occurred.
CPFAF88 E	Error occurred on call to MSF exit point program.
CPFAF8B E	Reserved mail message identifier is not valid.
CPFAF8D E	Reserved mail message identifier already used.

API Introduced: V3R1

List Mail Server Framework Configuration (QzmfLstMailCfg) API

Required Parameter Group:

1	Qualified user space name	Input	Char(20)
2	Format name	Input	Char(8)
3	Type configuration format name	Input	Char(8)
4	Type configuration	Input	Char(*)
5	Error code	I/O	Char(*)

Service Program: QZMFASRV

Threadsafe: No

The List Mail Server Framework Configuration (QzmfLstMailCfg) API generates a list of type configurations and places the list in a specified user space. You may specify a specific group of type configurations, a specific configuration or all configurations. The generated list replaces any existing data in the user space.

Authorities and Locks

User Space Authority

*CHANGE

Library Authority

*USE

User Space Lock

*EXCLRD

Required Parameter Group

Qualified user space name

INPUT; CHAR(20)

The qualified user space name identifies the space that is to receive the created list. The first 10 characters contain the user space name, and the second 10 characters contain the name of the library where the user space is located. You can use these special values for the library name:

**CURLIB*	The job's current library
**LIBL*	The library list

Format name

INPUT; CHAR(8)

The variable containing the name of the list format identifier. This must be set to LSTL0100. If the list format identifier is not LSTL0100, then the error message CPFAFB0 is generated.

Type configuration format name

INPUT; CHAR(8)

The format identifier of the parameter structure for the type configuration to be used. This must be set to LSTC0100. If the format identifier is not set to LSTC0100, the CPFAFB0 error message is generated.

Type configuration

INPUT; CHAR(*)

This is a structure containing the information that will be used for the mail server framework configuration being listed.

Error code

I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter. If this parameter is omitted, diagnostic and escape messages are issued to the application.

Format of the Generated List

The following table shows the layout of the parameter structure for LSTL0100. The framework configuration list consists of:

- A user area
- A generic header
- List data section

For detailed descriptions of the fields in the list returned, see Field Descriptions.

When you retrieve list entry information from a user space, you must use the entry size returned in the generic header. The size of each entry may be padded at the end. If you do not use the entry size, the result may not be valid.

LSTL0100 Format

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	CHAR(2)	Type group
2	2	CHAR(4)	Type value
6	6	CHAR(8)	Type name
14	E	CHAR(2)	Reserved
16	10	BINARY(4)	Type text coded character set identifier (CCSID)

20	14	CHAR(100)	Type text

Format for List Mail Configuration Selection

The following table shows the LSTC0100 format used to select the configurations to be listed. It can be used to selectively retrieve configuration entries. If the data entered in any of the fields is not valid, error message CPFAFB0 is generated. If no matching entries are found, not valid, error message CPFAFB1 is generated.

LSTC0100 Format

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	BINARY(4)	Length of structure
4	4	CHAR(2)	Selection type group
6	6	CHAR(4)	Selection type value
10	A	CHAR(8)	Selection type name

Field Descriptions

Length of structure. The length in bytes of the LSTL0100 format for list mail configuration selection.

Reserved. A reserved field must be set to blanks.

Selection type group. The requested type group to list. The possible values for the selection type group field are:

01	Address type group
02	Message type group
03	Envelope type group
04	Attachment type group
blank	List configurations that match the other selection criteria without using the selection type group as a selection criteria.

Selection type name. The name of the configuration to list. If the selection type name is set to blank, other selection criteria are used without using the selection type name as a selection criteria.

Selection type value. The value of the configuration to list. If the type value is set to blank, other selection criteria are used without using the selection type name as a selection criteria.

Type group. The type group of the entry that matched the selection criteria. The possible values for the type group field are:

01	Address type group
02	Message type group
03	Envelope type group

04	Attachment type group

Type name. The name of the type that matched the selection criteria. This is an 8-character mnemonic representation of the type. This value is made up of the characters A through Z and 0 through 9.

Type text. The description of the type that matched the selection criteria.

Type text coded character set identifier (CCSID). The coded character set identifier (CCSID) for the type text field.

Type value. The value of the type that matched the selection criteria. This is a 4-character representation of the type. This value is made up of the characters A through Z and 0 through 9.

Error Messages

Message ID	**Error Message Text**
CPF24B4 E	Severe error while addressing parameter list.
CPF3CF1 E	Error code parameter not valid.
CPF3C21 E	Format name &1 is not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPF3C36 E	Number of parameters, &1, entered for this API was not valid.
CPF3C90 E	Literal value cannot be changed.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.
CPFAFB0 E	Mail server framework configuration API error occurred.
CPFAFB1 E	Mail server framework type &1 not configured.

API Introduced: V3R1

Query Mail Message Identifier (QzmfQryMailMsgId) API

Required Parameter Group:

1	Mail message identifier	Input	Char(32)
2	Format name	Input	Char(8)
3	Status	Output	Char(1)
4	Error code	I/O	Char(*)

Service Program: QZMFASRV

Threadsafe: No

The Query Mail Message Identifier (QzmfQryMailMsgId) API is used to discover if a mail message identifier is known to the mail server framework.

Required Parameter Group

Mail message identifier

INPUT; CHAR(32)

The variable containing the unique message identifier that is being queried as to whether it is known by the mail server framework. The message identifier is composed of characters A through Z and 0 through 9 only.

Format name

INPUT; CHAR(8)

The variable containing the format identifier of the parameter list that is passed. This field must be set to QRYF0100.

Status

OUTPUT CHAR(1)

This field contains one of the following:

0	The mail message identifier is not known by the mail server framework. The message associated with this mail message may have already been processed.
1	The mail message identifier is known by the mail server framework. This status indicates that the mail message identifier was created using the Create Mail Message (QzmfCrtMailMsg) API. The message is either being processed, or is waiting to be processed, by the mail server framework.
2	The mail message identifier was reserved using the Reserve Mail Message Identifier (QzmfRsvMailMsgId) API. It was also created using the Create Mail Message (QzmfCrtMailMsg) API. The message still exists in the mail server framework and is either being processed, or is waiting to be processed, by the mail server framework.

3 The mail message identifier was reserved using the Reserve Mail Message Identifier (QzmfRsvMailMsgId) API. It was also created using the Create Mail Message (QzmfCrtMailMsg) API. The message no longer exists in the mail server framework. The message associated with this mail server identifier has already been processed by the mail server framework. However, the Creation Sequence Complete (QzmfCrtCmpMailMsg) API has not been called to remove the message identifier from the mail server framework.

4 The mail message identifier was reserved using the Reserve Mail Message Identifier (QzmfRsvMailMsgId) API. However, the Create Mail Message (QzmfCrtMailMsg) API has not been used to create a message using this mail message identifier.

Error code

I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3C90 E	Literal value cannot be changed.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.
CPFAF82 E	Error occurred during running of MSF API.
CPFAF83 E	Parameter error on call to MSF API. Reason code &1.
CPFAF84 E	MSF API failed. MSF message identifier not found.

API Introduced: V3R1

Remove Mail Server Framework Configuration (QzmfRmvMailCfg) API

Required Parameter Group:			
1	Type configuration	Input	Char(*)
2	Format name	Input	Char(8)
3	Error code	I/O	CHAR(*)

Service Program: QZMFASRV

Threadsafe: No

The Remove Mail Server Framework Configuration (QzmfRmvMailCfg) API deletes a configuration from the mail server framework.

Required Parameter Group

Type configuration

INPUT; CHAR(*)

A structure containing the information used for the mail server framework configuration being removed.

Format name

INPUT; CHAR(8)

The format identifier of the parameter structure for the type configuration being used. This must be set to DLTC0100.

Error code

I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter. If this parameter is omitted, diagnostic and escape messages are issued to the application.

DLTC0100 Format

The following table shows the layout of the parameter structure for DLTC0100. For a detailed description of each field, see Field Descriptions.

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	BIN(4)	Length of structure

4	4	CHAR(2)	Type group
6	6	CHAR(4)	Type value
10	A	CHAR(8)	Type name

Field Descriptions

Length of structure. The length in bytes of the parameter list format for the remove mail server framework configuration.

Type group. The type of the configuration being removed. This is a 2-character field representing the type group that is being configured. This field must be set to one of the allowed values. The allowed values are:

01	Address type group
02	Message type group
03	Envelope type group
04	Attachment type group

If the field is not set to an allowed value, the error message CPFAFB0 is generated.

Type name. The name of the type configuration being removed. This is an 8-character mnemonic representation of the type. This field must not be set to blanks. Allowed character values are A through Z and 0 through 9. If this field is blank, then error message CPFAFB0 is generated.

Type value. The value of the type configuration being removed. This is a 4-character representation of the type being removed. Allowed character values are A through Z and 0 through 9.

Error Messages

Message ID	**Error Message Text**
CPF24B4 E	Severe error while addressing parameter list.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPF3C36 E	Number of parameters, &1, entered for this API was not valid.
CPF3C90 E	Literal value cannot be changed.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.
CPFAFB0 E	Mail server framework configuration API error occurred.
CPFAFB1 E	Mail server framework type &1 not configured.

API Introduced: V3R1

Remove Reserved Mail Message Identifier (QzmfRmvRsvMailMsgId) API

Required Parameter Group:

1	Reserved mail message identifier	Input	Char(32)
2	Format name	Input	Char(8)
3	Error code	I/O	Char(*)

Service Program: QZMFASRV

Threadsafe: No

The Remove Reserved Mail Message Identifier (QzmfRmvRsvMailMsgId) API can be used to remove an identifier for an electronic mail message that has been reserved using the Reserve Mail Message (QzmfRsvMailMsgId) API but that has not been created. If the message has been created, the Creation Sequence Complete (QzmfCrtCmpMailMsg) API must be used to release the reserved message identifier. After the Remove Reserved Mail Message Identifier (QzmfRmvRsvMailMsgId) API completes successfully, the message identifier reserved earlier is no longer available to be used in message creation.

Required Parameter Group

Reserved mail message identifier

INPUT; CHAR(32)

The variable containing the unique message identifier to be removed from reserved status. After this API has completed running, the message identifier can no longer be used with the Create Mail Message (QzmfCrtMailMsg) API to pass information that defines the message. The reserved message identifier is composed of characters A through Z and 0 through 9 only.

Format name

INPUT; CHAR(8)

The variable containing the format identifier of the parameter list being passed. This field must be set to RMVF0100.

Error code

I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3C90 E	Literal value cannot be changed.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.
CPFAF82 E	Error occurred during running of MSF API.
CPFAF83 E	Parameter error on call to MSF API. Reason code &1.
CPFAF8B E	Reserved mail message identifier is not valid.
CPFAF8D E	Reserved mail message identifier already used.

API Introduced: V3R1

Reserve Mail Message Identifier (QzmfRsvMailMsgId) API

Required Parameter Group:

1	Reserved mail message identifier	Output	Char(32)
2	Format name	Input	Char(8)
3	Error code	I/O	Char(*)

Service Program: QZMFASRV

Threadsafe: No

The Reserve Mail Message Identifier (QzmfRsvMailMsgId) API is used to reserve an identifier for an electronic mail message. After the API completes successfully, a single mail message identifier is returned to the caller.

This reserved mail message identifier can be operated on through the Remove Reserved Mail Message Identifier (QzmfRmvRsvMailMsgId) API, the Creation Sequence Complete (QzmfCrtCmpMailMsg) API, and the Create Mail Message (QzmfCrtMailMsg) API. The normal processing for a reserved mail message identifier involves the following sequence:

1. The mail message identifier is reserved (using the Reserve Mail Message Identifier (QzmfRsvMailMsgId) API). This means that a message can later be created using this mail message identifier.
2. The message is created (using the Create Mail Message (QzmfCrtMailMsg) API).
3. The application confirms that the message was created (using the Creation Sequence Complete (QzmfCrtCmpMailMsg) API).

A reserved mail message identifier can be removed from reserved status using the remove mail message identifier if the message has not been created.

Required Parameter Group

Reserved mail message identifier

OUTPUT; CHAR(32)

The variable containing a unique message identifier. The message identifier is returned by the API after it has completed running. This identifier can be used later by the Create Mail Message (QzmfCrtMailMsg) API to pass information which defines the message. The message identifier is composed of characters A through Z and 0 through 9 only. If an error occurs during processing, a message identifier is not generated.

Format name

INPUT; CHAR(8)

The variable containing the format identifier of the parameter list being passed. This field must be set to RSVF0100.

Error code

I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3C90 E	Literal value cannot be changed.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.
CPFAF82 E	Error occurred during running of MSF API.
CPFAF83 E	Parameter error on call to MSF API. Reason code &1.

API Introduced: V3R1

Top | Office APIs | APIs by category

Retrieve Mail Message (QzmfRtvMailMsg) API

Required Parameter Group:

1	Mail message identifier	Input	Char(32)
2	Message descriptor attributes	Input	Char(*)
3	Number of message descriptor attributes	Input	Array of Binary(4)
4	Format name	Input	CHAR(8)
5	Error code	I/O	Char(*)

Service Program: QZMFASRV

Threadsafe: No

The Retrieve Mail Message (QzmfRtvMailMsg) API retrieves information about an electronic mail message and returns it in the receiver variables provided by the caller. It is only valid to call this API during the processing of the exit program for the Snap-In Call or Track Mail Message Changes exit point.

Required Parameter Group

Mail message identifier

INPUT; CHAR(32)

The variable containing the message identifier for which information is to be retrieved. The mail message identifier is composed of characters A through Z and 0 through 9 only.

Message descriptor attributes

INPUT; ARRAY OF CHAR(*)

This array contains the message descriptor attributes for the data that is to be retrieved. The following table defines a message descriptor attributes array entry.

Table 1. Message Descriptor Attribute Array Entry

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	POINTER	Receiver message descriptor pointer
16	10	BINARY(4)	Length of the receiver message descriptor
20	14	CHAR(8)	Format name
28	1C	BINARY(4)	Reserved (must be set to zero)

The message descriptor attribute array entry contains the following:

- Pointers to the message descriptors that are to receive the message information
- The format name associated with the message descriptor

- The length of the message descriptor.

If the message descriptors are not long enough to hold all of the available message information, the data is truncated and the number of bytes available is returned. If the length specified for the receiver message descriptor is -1, the pointer to the message descriptor is pointing to a user space that should be automatically extended by the API if it is not large enough for all of the data.

The format name associated with the message descriptor indicates what type of message information is being requested.

Number of message descriptor attributes

INPUT; BINARY(4)

The number of entries in the array of message descriptor attributes. At least one array element must be specified on a call to this API.

Format name

INPUT; CHAR(8)

The variable containing the format name of the parameter list being returned. This field must be set to RTVM0100.

Error code

I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter.

Message Parameter Lists

A message descriptor consists of:

- A common message header section
- The actual message descriptor format. The following table shows valid message descriptors.

Format name	Description
ORCL0100	Originator recipient entry
ORGL0100	Originator entry
ENVL0100	Envelope entry
RCPL0100	Recipient entry
RCHL0100	Recipient history entry
ROAL0100	Report-on address entry
RPYL0100	Reply-to address entry
RTAL0100	Report-to address entry
ATTL0100	Attachment reference entry
MSGL0100	Message types in the recipient list entries
EXCH0100	Exit call history entry

CRTA0100	Creation attributes entry

Message Descriptor Common Header

The following table shows the common header section that is always included at the beginning of a message descriptor. It is used for all of the different formats that can be retrieved using this API. The common header will always be returned, even if there is no message descriptor data to be included. There may be no list items or one or more list items after the common header section.

Table 2. Format of Common Header

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	BINARY(4)	Length of this message descriptor
4	4	BINARY(4)	Number of bytes available for this message descriptor
8	8	CHAR(8)	Array element format name
16	10	BINARY(4)	Offset of the first entry in the message descriptor
20	14	BINARY(4)	Number of entries available for this message descriptor
24	18	BINARY(4)	Reserved (must be set to zero)
*	*	CHAR(*)	Message descriptor data

Message Descriptor Formats for Retrieve Mail Message

If there is data after the common header, the format associated with the message descriptor is used. There may be multiple instances of the formats following the common header, depending on the size of the parameter list.

ORCL0100 Format (Original Recipient Entry)

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	BINARY(4)	Length of this entry
4	4	BINARY(4)	Displacement of the original recipient address from the beginning of this ORCL0100 entry
8	8	BINARY(4)	Length of original recipient address
12	C	CHAR(4)	Address type
16	10	BINARY(4)	Original recipient address coded character set identifier(CCSID)
20	14	BINARY(4)	Distribution type
24	18	BINARY(4)	Reply requested flag
28	1C	BINARY(4)	Unique identifier

32	20	BINARY(4)	Unique identifier of referenced ORCL0100 entry
36	24	BINARY(4)	Reserved (must be set to zero)
*	*	CHAR(*)	Original recipient address

ORGL0100 Format (Originator Entry)

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	BINARY(4)	Length of this entry
4	4	BINARY(4)	Displacement of the origin address from the beginning of this ORGL0100 entry
8	8	BINARY(4)	Length of origin address
12	C	CHAR(4)	Address type
16	10	BINARY(4)	Origin address coded character set identifier(CCSID)
20	14	BINARY(4)	Unique identifier
24	18	BINARY(4)	Unique identifier of referenced ORGL0100 entry
28	1C	BINARY(4)	Reserved (set to zero)
*	*	CHAR(*)	Origin address

ENVL0100 Format (Envelope Entry)

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	BINARY(4)	Length of this entry
4	4	BINARY(4)	Displacement of envelope from the beginning of this ENVL0100 entry
8	8	BINARY(4)	Length of envelope
12	C	CHAR(4)	Envelope type
16	10	BINARY(4)	Unique identifier
20	14	BINARY(4)	Unique identifier of referenced ENVL0100 entry
24	18	BINARY(4)	Reserved (set to zero)
*	*	CHAR(*)	Envelope

RCPL0100 Format (Recipient Entry)

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Length of this entry
4	4	BINARY(4)	Displacement of the snap-in provided information (SPIN) from the beginning of this RCPL0100 entry
8	8	BINARY(4)	Length of snap-in provided information (SPIN)
12	C	BINARY(4)	Displacement of the recipient address from the beginning of this RCPL0100 entry
16	10	BINARY(4)	Length of recipient address
20	14	CHAR(4)	Address type
24	18	BINARY(4)	Recipient address coded character set identifier (CCSID)
28	1C	BINARY(4)	Reason code
32	20	BINARY(4)	Diagnostic code
36	24	CHAR(4)	Message type
40	28	BINARY(4)	Status
44	2C	BINARY(4)	Reserved
48	30	BINARY(4)	Unique identifier
52	34	BINARY(4)	Reserved (set to zero)
*	*	CHAR(*)	Recipient address
*	*	CHAR(*)	Snap-in provided information (SPIN)

ROAL0100 Format (Report-on Address Entry)

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Length of this entry
4	4	BINARY(4)	Displacement of the snap-in provided information (SPIN) from the beginning of this ROAL0100 entry
8	8	BINARY(4)	Length of snap-in provided information (SPIN)
12	C	BINARY(4)	Displacement of the report-on address from the beginning of this ROAL0100 entry
16	10	BINARY(4)	Length of address
20	14	CHAR(4)	Address type
24	18	BINARY(4)	Report-on address coded character set identifier (CCSID)
28	1C	BINARY(4)	Reason code
32	20	BINARY(4)	Diagnostic code

36	24	BINARY(4)	Unique identifier
40	28	BINARY(4)	Unique identifier of referenced ROAL0100 entry
44	2C	BINARY(4)	Reserved (set to zero)
*	*	CHAR(*)	Report-on address
*	*	CHAR(*)	Snap-in provided information (SPIN)

RPYL0100 Format (Reply-to Address Entry)

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	BINARY(4)	Length of this entry
4	4	BINARY(4)	Displacement of the reply-to address from the beginning of this RPYL0100 entry
8	8	BINARY(4)	Length of reply-to address
12	C	CHAR(4)	Address type
16	10	BINARY(4)	Reply-to address coded character set identifier (CCSID)
20	14	BINARY(4)	Unique identifier
24	18	BINARY(4)	Unique identifier of referenced RPYL0100 entry
28	1C	BINARY(4)	Reserved (must be set to zero)
*	*	CHAR(*)	Reply-to address

RTAL0100 Format (Report-to Address Entry)

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	BINARY(4)	Length of this entry
4	4	BINARY(4)	Displacement of the report-to address from the beginning of this RTAL0100 entry
8	8	BINARY(4)	Length of address
12	C	CHAR(4)	Address type
16	10	BINARY(4)	Report-to address coded character set identifier (CCSID)
20	14	BINARY(4)	Unique identifier
24	18	BINARY(4)	Unique identifier of referenced RTAL0100 entry
28	1C	BINARY(4)	Reserved (set to zero)
*	*	CHAR(*)	Report-to address

ATTL0100 Format (Attachment Reference Entry)

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Length of this entry
4	4	BINARY(4)	Displacement of the attachment reference from the beginning of this ATTL0100 entry
8	8	BINARY(4)	Length of attachment reference
12	C	CHAR(4)	Attachment reference type
16	10	BINARY(4)	Unique identifier
20	14	BINARY(4)	Unique identifier of referenced ATTL0100 entry
24	18	BINARY(4)	Reserved (set to zero)
*	*	CHAR(*)	Attachment reference

MSGL0100 Format (Message Types in the Recipient List Entry)

Offset		Type	Field
Dec	Hex		
0	0	CHAR(4)	Message type

RCHL0100 Format (Recipient History Entry)

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Length of this entry
4	4	BINARY(4)	Displacement of the snap-in provided information (SPIN) of this RCHL0100 entry
8	8	BINARY(4)	Length of snap-in provided information (SPIN)
12	C	BINARY(4)	Displacement of the recipient address from the beginning of this RCHL0100 entry
16	10	BINARY(4)	Length of recipient address
20	14	CHAR(4)	Address type
24	18	BINARY(4)	Recipient address coded character set identifier (CCSID)
28	1C	CHAR(4)	Message type
32	20	BINARY(4)	Status
36	24	BINARY(4)	Reserved

40	28	BINARY(4)	Unique identifier
44	2C	BINARY(4)	Recipient status flag
48	30	BINARY(4)	Unique identifier of parent entry
52	34	BINARY(4)	Reserved (set to zero)
*	*	CHAR(*)	Recipient address
*	*	CHAR(*)	Snap-in provided information (SPIN)

EXCH0100 Format (Exit Point Call History Entry)

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	CHAR(20)	Mail server framework exit point name
20	14	CHAR(10)	Qualified mail server framework exit program name
30	1E	CHAR(10)	Mail server framework exit program library
40	28	BINARY(4)	Exit program number
44	2C	CHAR(16)	Timestamp of when the exit program was called
60	3C	CHAR(16)	Timestamp of when the exit program returned
76	4C	BINARY(4)	Return code from the exit program
80	50	CHAR(1)	Change indicator
81	51	CHAR(3)	Reserved (set to blanks)

CRTA0100 Format (Creation Attributes Entry)

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	CHAR(16)	Timestamp of when the message was created
16	10	CHAR(4)	Creation message type

Field Descriptions

Address type. The type of address that is contained in the entry.

Attachment reference. A reference to an attachment that is associated with the message.

Attachment reference type. The type of attachment that is contained in the entry.

Change indicator. An indicator of whether processing of the exit point program resulted in changes to the message.

0	There were no changes to the message as a result of this exit point program being called.
1	There were changes to the message as a result of this exit point program being called.

Creation message type. The creation message type specified when the MSF message was created.

Diagnostic code. A code that indicates the cause of a problem which caused the delivery of a message to this address to fail. The diagnostic code is assumed to contain the X.400 nondelivery diagnostic code.

Displacement of the attachment reference from the beginning of this ATTL0100 entry. The displacement from the beginning of this entry to the attachment reference in this entry.

Displacement of the envelope from the beginning of this ENVL0100 entry. The displacement from the beginning of this entry to the envelope in this entry.

Displacement of the origin address from the beginning of this ORGL0100 entry. The displacement from the beginning of this entry to the origin address in this entry.

Displacement of the original recipient address from the beginning of this ORCL0100 entry. The displacement from the beginning of this entry to the original recipient address in this entry.

Displacement of the recipient address from the beginning of this RCPL0100 entry. The displacement from the beginning of this entry to the recipient address in this entry.

Displacement of the reply-to address from the beginning of this RPYL0100 entry. The displacement from the beginning of this entry to the reply-to address in this entry.

Displacement of the report-to address from the beginning of this RTAL0100 entry. The displacement from the beginning of this entry to the report-to address in this entry.

Displacement of the report-on address from the beginning of this ROAL0100 entry. The displacement from the beginning of this entry to the report-on address in this entry.

Displacement of the snap-in provided information (SPIN) of this RCHL0100. The displacement from the beginning of this entry to the snap-in provided information (SPIN). (There are fields for RCPL0100 and RCHL0100 entries.)

Distribution type. The type of distribution associated with each recipient entry. The possible values are:

0	Normal message recipient
1	CC (carbon copy) recipient
2	BCC (blind carbon copy) recipient

Envelope. A string of data representing information about the message, aside from the attachments and its recipients.

Envelope type. The type of envelope that is contained in the entry.

Exit program number. A number assigned to the exit program when it is registered using the registration facility. This is the number in effect at the time the exit program was called.

Format name. The content and format of the information provided for each message parameter. The possible values are:

ORCL0100	Original recipient entry

ORGL0100	Originator entry
ENVL0100	Envelope entry
RCPL0100	Recipient entry
RCHL0100	Recipient history entry
ROAL0100	Report-on address entry
RPYL0100	Reply-to address entry
RTAL0100	Report-to address entry
ATTL0100	Attachment reference entry
MSGL0100	List of message types in the recipient list entry
EXCH0100	Exit call history entry
CRTA0100	Creation attributes entry

Length of address. The length of address that is contained in the entry. The maximum length of an address is 1024 bytes.

Length of attachment reference. The length in bytes of the attachment reference that is contained in the entry.

Length of envelope. The length of envelope that is contained in the entry.

Length of origin address. The length of the origin address for this entry. The maximum length of an origin address is 1024 bytes.

Length of original recipient address. The length of the original recipient address for this entry. The maximum length of an original recipient address is 1024 bytes.

Length of receiver message descriptor. The length in bytes of the message descriptor that is being pointed to by the pointer to the message descriptor. The maximum length of a message descriptor is 16 million bytes.

Length of recipient address. The length in bytes of the recipient address. The maximum length of a recipient address is 1024 bytes.

Length of reply-to address. The length in bytes of the reply-to address. The maximum length of a reply-to address is 1024 bytes.

Length of snap-in provided information (SPIN). The length in bytes of the snap-in provided information (SPIN). The maximum length of the SPIN is 256 bytes.

Length of this entry. The length in bytes of this entry. This is used to get to the next entry.

Length of this message descriptor. The length in bytes of this message descriptor. The maximum length of a message descriptor is 16 million bytes.

Message descriptor data. One or more parameter list formats that follow the common header. Message descriptors are made up of a common header and a list of entries. The format of each entry in the list is defined by the format name associated with the message descriptor, which is located in the common header. The number of entries in the list is also defined in the common header.

Message type. The type of message that is associated with the entry.

Number of bytes available for this message descriptor. The number of bytes available in the space where the mail server framework puts the information being retrieved.

Number of entries available for this message descriptor. The total number of entries available for this message descriptor indicates the number of entries that would be returned if the length of the message descriptor is greater than or equal to the number of bytes available for this message descriptor.

Offset of the first entry in the message descriptor. The offset from the beginning of this message descriptor to the first entry in the list of entries.

Origin address. A string that represents the address associated with the originator of the message. The contents and format of the string are not defined by the mail server framework. It is assumed that the address type associated with the origin address defines the contents of the origin address field.

Origin address coded character set identifier (CCSID). The CCSID provided for the origin address. Valid values for the CCSID are 1 through 65533 and 65535.

Original recipient address. A string that represents the address associated with the original recipient of the message. The contents and format of the string are not defined by the mail server framework. It is assumed that the address type associated with the original recipient address defines the contents of the original recipient address field.

Original recipient address coded character set identifier (CCSID). The CCSID provided for the original recipient address. Valid values for the CCSID are 1 through 65533 and 65535.

Qualified mail server framework exit program name. The qualified program name of the program that was called at the mail server framework exit point.

Reason code. A code that identifies reasons associated with the message delivery to this address. In the case of a nondelivery entry, this field would contain the reason the delivery of this message to this recipient failed. The reason code is assumed to contain the X.400 nondelivery reason code.

Receiver message descriptor pointer. A space pointer that points to a message descriptor. A message descriptor is used to pass information which describes the message.

Recipient address. A string that represents the address associated with a recipient of the message. The contents and format of the string are not defined by the mail server framework. It is assumed that the address type associated with the recipient address defines the contents of the recipient address field.

Recipient address coded character set identifier (CCSID). The CCSID provided for the recipient address. Valid values for the CCSID are 1 through 65533 and 65535.

Recipient status flag. A flag which when set to 1 indicates that this entry has been replaced by either one or multiple entries. Entries with this flag set to 1 are referred to as parents. Entries with this flag set zero are referred to as children.

Reply requested flag. Whether this original recipient should reply to the message. The possible values are as follows:

0	A reply is not requested from this original recipient
1	A reply is requested from this original recipient

Reply-to address. A string that represents the address to be replied to. The contents and format of the string are not defined by the mail server framework. It is assumed that the address type associated with the reply-to address defines the contents of the reply-to address field.

Reply-to address coded character set identifier (CCSID). The CCSID provided for the reply-to address. Valid values for the CCSID are 1 through 65533 and 65535.

Report-on address. A string which represents the address to be reported on. The contents and format of the string are not defined by the mail server framework. It is assumed that the address type associated with the report-on address defines the contents of the report-on address field.

Report-on address coded character set identifier (CCSID). The CCSID provided for the report-on address. Valid values for the CCSID are 1 through 65533 and 65535.

Report-to address. A string which represents the address to be reported to. The contents and format of the string are not defined by the mail server framework. It is assumed that the address type associated with the report-to address defines the contents of the report-to address field.

Report-to address coded character set identifier (CCSID). The CCSID provided for the report-to address. Valid values for the CCSID are 1 through 65533 and 65535.

Reserved. All reserved fields must be set to zero.

Return code from the exit program. The return code returned by the user exit program.

Snap-in provided information (SPIN). An area where snap-ins can store information that other snap-ins may use. SPIN provides a place where information relating to a specific recipient can be stored and used by snap-ins in the same or different exit points. This is completely user-defined and user-interpreted data.

Status. The status associated with each recipient entry. The possible values are:

1	Forwarded (remote)
2	Ignore
3	Local
4	Nondeliverable
5	Security violation

Timestamp of when the message was created. The timestamp of when the message was created. The format of the timestamp is CYYMMDDHHMMSSMMM. The fields are defined in the order that they occur in the string.

C	Century, where 0 indicates years 19*xx* and 1 indicates years 20*xx*.
YY	Year
MM	Month
DD	Day
HH	Hour of the day
mm	Minute of the hour
SS	Second of the minute
MMM	Milliseconds in zoned format

Timestamp of when the exit program was called. The timestamp of when the exit point was called. The format of this timestamp is the same as that of the timestamp of when the message was created.

Timestamp of when the exit program returned. The timestamp of when the exit point returned. The format of this timestamp is the same as that of the timestamp of when the message was created.

Unique identifier. A unique identifier that differentiates each item within a particular list. Identifiers are generated for each list item when the Create Mail Message (QzmfCrtMailMsg) API has successfully completed. These unique identifiers are temporary and may change as the mail service processes a message. List entries are placed in message descriptors such that unique identifiers are in ascending order.

Unique identifier of parent entry. The unique identifier associated with the parent entry of this entry. A parent entry is an entry that had been replaced by a single or multiple entries.

Unique identifier of referenced entry. The unique identifier of another ORCL0100, ORGL0100, ENVL0100, RTAL0100, ROAL0100, RPYL0100, or ATTL0100 entry that this entry refers to. This field can be used to create entry cross-references as new entries are added.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3C90 E	Literal value cannot be changed.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.
CPFAF80 E	Syntax error on call to MSF API. Reason code &1.
CPFAF82 E	Error occurred during running of MSF API.
CPFAF83 E	Parameter error on call to MSF API. Reason code &1.
CPFAF84 E	MSF API failed. MSF message identifier not found.
CPFAF85 E	MSF API failed. Request not allowed.

API Introduced: V3R1

Snap-In Call Exit Program

Required Parameter Group:

1	Exit point name	Input	Char(20)
2	Mail message identifier	Input	Char(32)
3	Message descriptor attributes	Input	Array of Char(*)
4	Number of message descriptor attributes	Input	Binary(4)
5	Format name	Input	Char(8)
6	Return code	Output	Binary(4)

The Snap-In Call exit program is used to pass information about an electronic mail message to defined snap-in programs. When the snap-in program has completed processing the message, it returns its status in the return code parameter.

Parameter Group

Exit point name

INPUT; CHAR(20)

The variable containing the name of the user exit point that is calling the snap-in program. The following table shows the exit points from which snap-in programs can be called.

Exit Point Name	Description
QIBM_QZMFMSF_LST_EXP	MSF List Expansion Exit
QIBM_QZMFMSF_ADR_RSL	MSF Address Resolution Exit
QIBM_QZMFMSF_ENL_PSS	MSF Envelope Processing Exit
QIBM_QZMFMSF_ATT_CNV	MSF Attachment Conversion Exit
QIBM_QZMFMSF_SEC_AUT	MSF Security and Authority Exit
QIBM_QZMFMSF_LCL_DEL	MSF Local Delivery Exit
QIBM_QZMFMSF_MSG_FWD	MSF Messaging Forwarding Exit
QIBM_QZMFMSF_NON_DEL	MSF Non Delivery Exit
QIBM_QZMFMSF_ATT_MGT	MSF Attachment Management Exit
QIBM_QZMFMSF_ACT	MSF Accounting User Exit

Mail message identifier

INPUT; CHAR(32)

The variable containing the message identifier that the Snap-In Call exit is passing information for. The mail message identifier is composed of characters A through Z and 0 through 9 only.

Message descriptor attributes

INPUT; ARRAY OF CHAR(*)

This array contains the message descriptor attributes for the data that is to be passed. The following table defines a message descriptor attributes array entry.

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	POINTER	Message descriptor pointer
16	10	BINARY(4)	Length of the message descriptor
20	14	CHAR(8)	Message descriptor format name
28	1C	BINARY(4)	Reserved (must be set to zero)

Number of message descriptor attributes

INPUT; BINARY(4)

The number of message descriptor attribute entries being passed in the array.

Format name

INPUT; CHAR(8)

The variable containing the format identifier of the parameter list in which information is being passed. This field is set to SPCL0100. This is defined in the first 8 characters of the exit program data associated with the exit point programs that are registered to work with the exit points from which snap-in programs can be called.

Return code

OUTPUT; BINARY(4)

The defined return codes are:

0	Continue processing
1	Backout changes made but continue processing.
2	End the mail server framework job.
3	End the processing of this message.

Message Descriptors

A message descriptor consists of:

- A snap-in call message header section
- A message descriptor data section

  This section is composed of a list of individual entries. The data in these entries is in one of the formats listed below. The following table shows valid array element format names for the data list section:

Format Name	Description
ORCL0100	Original recipient entry
ORGL0100	Originator entry
ENVL0100	Envelope entry
RCPL0100	Recipient entry
ROAL0100	Report-on address entry
RPYL0100	Reply-to address entry
RTAL0100	Report-to address entry
ATTL0100	Attachment reference entry
MSGL0100	Message types in the recipient list entry
CRTA0100	Creation attributes entry

Not all of these formats are passed to the Snap-In Call exit at each of the user exit points. The following table identifies the format of the message descriptors that are passed at each exit point.

Exit Point Name	Formats Passed
QIBM_QZMFMSF_LST_EXP	RCPL0100 ENVL0100
QIBM_QZMFMSF_ADR_RSL	RCPL0100 ENVL0100 ORGL0100 RTAL0100 ROAL0100 RPYL0100 ORCL0100
QIBM_QZMFMSF_ENL_PSS	MSGL0100 RCPL0100 ROAL0100 ENVL0100 ATTL0100 RTAL0100 RPYL0100 ORCL0100 ORGL0100

QIBM_QZMFMSF_ATT_CNV	MSGL0100 RCPL0100 ATTL0100 ENVL0100
QIBM_QZMFMSF_SEC_AUT	RCPL0100 ATTL0100 ORGL0100
QIBM_QZMFMSF_LCL_DEL	RCPL0100 ENVL0100 ATTL0100 ORGL0100 ROAL0100 RTAL0100 RPYL0100 ORCL0100 CRTA0100
QIBM_QZMFMSF_MSG_FWD	RCPL0100 ENVL0100 ATTL0100 ORGL0100 ROAL0100 RTAL0100 RPYL0100 ORCL0100 CRTA0100
QIBM_QZMFMSF_NON_DEL	RCPL0100 ENVL0100 ATTL0100 ORGL0100 RTAL0100

QIBM_QZMFMSF_ATT_MGT	RCPL0100 ATTL0100
QIBM_QZMFMSF_ACT	RCPL0100 ENVL0100 ATTL0100 ORGL0100 RTAL0100 RPYL0100 ORCL0100 CRTA0100

Snap-In Call Common Header

The following table shows the common header section that is always included at the beginning of a message descriptor. It is used for all of the different formats that can be passed when calling Snap-In Call exit programs. The common header will always be passed when there is message descriptor data to be included. The message descriptor data may include one or more individual entries after the common header section.

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	BINARY(4)	Length of this message descriptor
4	4	BINARY(4)	Reserved (must be set to zero)
8	8	CHAR(8)	Message descriptor format name
16	10	BINARY(4)	Offset of the first entry in the message descriptor
20	14	BINARY(4)	Number of entries in this message descriptor
24	18	BINARY(4)	Reserved (must be set to zero)
*	*	CHAR(*)	Message descriptor data. See the sections that follow.

Message Descriptor Formats for Snap-In Call

If there is data after the common header, the format associated with the message descriptor is used.

ORCL0100 Format (Original Recipient Entry)

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Length of this entry
4	4	BINARY(4)	Displacement of the original recipient address from the beginning of this ORCL0100 entry
8	8	BINARY(4)	Length of original recipient address
12	C	CHAR(4)	Address type
16	10	BINARY(4)	Original recipient address coded character set identifier(CCSID)
20	14	BINARY(4)	Distribution type
24	18	BINARY(4)	Reply requested flag
28	1C	BINARY(4)	Unique identifier
32	20	BINARY(4)	Unique identifier of referenced ORCL0100 entry
36	24	BINARY(4)	Reserved (must be set to zero)
*	*	CHAR(*)	Original recipient address

ORGL0100 Format (Originator Entry)

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Length of this entry
4	4	BINARY(4)	Displacement of the origin address from the beginning of this ORGL0100 entry
8	8	BINARY(4)	Length of origin address
12	C	CHAR(4)	Address type
16	10	BINARY(4)	Origin address coded character set identifier (CCSID)
20	14	BINARY(4)	Unique identifier
24	18	BINARY(4)	Unique identifier of referenced ORGL0100 entry
28	1C	BINARY(4)	Reserved (must be set to zero)
*	*	CHAR(*)	Origin address

ENVL0100 Format (Envelope Entry)

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Length of this entry

4	4	BINARY(4)	Displacement of envelope from the beginning of this ENVL0100 entry
8	8	BINARY(4)	Length of envelope
12	C	CHAR(4)	Envelope type
16	10	BINARY(4)	Unique identifier
20	14	BINARY(4)	Unique identifier of referenced ENVL0100 entry
24	18	BINARY(4)	Reserved (must be set to zero)
*	*	CHAR(*)	Envelope

RCPL0100 Format (Recipient Entry)

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	BINARY(4)	Length of this entry
4	4	BINARY(4)	Displacement of the snap-in provided information (SPIN) from the beginning of this RCPL0100 entry
8	8	BINARY(4)	Length of snap-in provided information (SPIN)
12	C	BINARY(4)	Displacement of the recipient address from the beginning of this RCPL0100 entry
16	10	BINARY(4)	Length of recipient address
20	14	CHAR(4)	Address type
24	18	BINARY(4)	Recipient address coded character set identifier (CCSID)
28	1C	BINARY(4)	Reason code
32	20	BINARY(4)	Diagnostic code
36	24	CHAR(4)	Message type
40	28	BINARY(4)	Status
44	2C	BINARY(4)	Reserved
48	30	BINARY(4)	Unique identifier
52	34	BINARY(4)	Reserved (must be set to zero)
*	*	CHAR(*)	Recipient address
*	*	CHAR(*)	Snap-in provided information (SPIN)

RPYL0100 Format (Reply-to Address Entry)

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	BINARY(4)	Length of this entry

4	4	BINARY(4)	Displacement of the reply-to address from the beginning of this RPYL0100 entry
8	8	BINARY(4)	Length of reply-to address
12	C	CHAR(4)	Address type
16	10	BINARY(4)	Reply-to address coded character set identifier (CCSID)
20	14	BINARY(4)	Unique identifier
24	18	BINARY(4)	Unique identifier of referenced RPYL0100 entry
28	1C	BINARY(4)	Reserved (must be set to zero)
*	*	CHAR(*)	Reply-to address

ROAL0100 Format (Report-on-Address Entry)

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	BINARY(4)	Length of this entry
4	4	BINARY(4)	Displacement of the snap-in provided information (SPIN) from the beginning of this ROAL0100 entry
8	8	BINARY(4)	Length of snap-in provided information (SPIN)
12	C	BINARY(4)	Displacement of the report-on address from the beginning of this ROAL0100 entry
16	10	BINARY(4)	Length of address
20	14	CHAR(4)	Address type
24	18	BINARY(4)	Report-on address coded character set identifier (CCSID)
28	1C	BINARY(4)	Reason code
32	20	BINARY(4)	Diagnostic code
36	24	BINARY(4)	Unique identifier
40	28	BINARY(4)	Unique identifier of referenced ROAL0100 entry
44	2C	BINARY(4)	Reserved
*	*	CHAR(*)	Report-on address
*	*	CHAR(*)	Snap-in provided information (SPIN)

RTAL0100 Format (Report-to Address Entry)

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	BINARY(4)	Length of this entry

4	4	BINARY(4)	Displacement of the report-to address from the beginning of this RTAL0100 entry
8	8	BINARY(4)	Length of address
12	C	CHAR(4)	Address type
16	10	BINARY(4)	Report-to address coded character set identifier (CCSID)
20	14	BINARY(4)	Unique identifier
24	18	BINARY(4)	Unique identifier of referenced RTAL0100 entry
28	1C	BINARY(4)	Reserved (must be set to zero)
*	*	CHAR(*)	Report-to address

ATTL0100 Format (Attachment Reference Entry)

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	BINARY(4)	Length of this entry
4	4	BINARY(4)	Displacement of the attachment reference from the beginning of this ATTL0100 entry
8	8	BINARY(4)	Length of attachment reference
12	C	CHAR(4)	Attachment reference type
16	10	BINARY(4)	Unique identifier
20	14	BINARY(4)	Unique identifier of referenced ATTL0100 entry
24	18	BINARY(4)	Reserved (must be set to zero)
*	*	CHAR(*)	Attachment reference

MSGL0100 Format (Message Type in the Recipient List Entry)

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	CHAR(4)	Message type

CRTA0100 Format (Creation Attributes Entry)

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	CHAR(16)	Timestamp of when the message was created
16	10	CHAR(4)	Creation message type

Field Descriptions

Address type. The type of address that is contained in the entry.

Attachment reference. A reference to an attachment that is associated with the message.

Attachment reference type. The type of attachment that is contained in the entry.

Creation message type. The creation message type specified when the MSF message was created.

Diagnostic code. A code that indicates the cause of a problem which caused the delivery of a message to this address to fail. The diagnostic code is assumed to contain the X.400 nondelivery diagnostic code.

Note: The reason code and diagnostic code fields are assumed to contain values that are documented in *CCITT Data Communication Networks Message Handling Systems 1988 Recommendation for X.400-X.420* for fields Non-delivery-reason-code and Non-delivery-diagnostic-code.

Displacement of the attachment reference from the beginning of this ATTL0100 entry. The displacement from the beginning of this entry to the attachment reference in this entry.

Displacement of the envelope from the beginning of this ENVL0100 entry. The displacement from the beginning of this entry to the envelope in this entry.

Displacement to the original recipient address from the beginning of this ORCL0100 entry. The displacement from the beginning of this entry to the original recipient address in this entry.

Displacement of the origin address from the beginning of this ORGL0100 entry. The displacement from the beginning of this entry to the origin address in this entry.

Displacement of the recipient address from the beginning of this RCPL0100 entry. The displacement from the beginning of this entry to the recipient address in this entry.

Displacement of the reply-to address from the beginning of this RPYL0100 entry. The displacement from the beginning of this entry to the reply-to address in this entry.

Displacement of the report-to address from the beginning of this RTAL0100 entry. The displacement from the beginning of this entry to the report-to address in this entry.

Displacement of the report-on address from the beginning of this ROAL0100 entry. The displacement from the beginning of this entry to the report-on address in this entry.

Displacement of the snap-in provided information (SPIN) of this RCHL0100. The displacement from the beginning of this entry to the snap-in provided information (SPIN). (There are fields for RCPL0100 and RCHL0100 entries.)

Distribution type. The type of distribution associated with each recipient entry. The possible values are:

0	Normal message recipient
1	CC (carbon copy) recipient
2	BCC (blind carbon copy) recipient

Envelope. A string of data representing information about the message, aside from the attachments and its recipients.

Envelope type. The type of envelope that is contained in the entry.

Exit program number. A number assigned to the exit program when it is registered using the registration facility. This is the number in effect at the time the exit program was called.

Format name. The content and format of the information provided for each message parameter. The possible values are:

ORCL0100	Original recipient entry
ORGL0100	Originator entry
ENVL0100	Envelope entry
RCPL0100	Recipient entry
ROAL0100	Report-on address entry
RPYL0100	Reply-to address entry
RTAL0100	Report-to address entry
ATTL0100	Attachment reference entry
MSGL0100	List of message types in the recipient list entry

Length of address. The length of address that is contained in the entry. The maximum length of an address is 1024 bytes.

Length of attachment reference. The length in bytes of the attachment reference that is contained in the entry.

Length of envelope. The length of envelope that is contained in the entry.

Length of origin address. The length of the origin address for this entry. The maximum length of an origin address is 1024 bytes.

Length of original recipient address. The length of the original recipient address for this entry. The maximum length of an original recipient address is 1024 bytes.

Length of receiver message descriptor. The length in bytes of the message descriptor that is being pointed to by the pointer to the message descriptor. The maximum length of a message descriptor is 16 million bytes.

Length of recipient address. The length in bytes of the recipient address. The maximum length of a recipient address is 1024 bytes.

Length of reply-to address. The length in bytes of the reply-to address. The maximum length of a reply-to address is 1024 bytes.

Length of snap-in provided information (SPIN). The length in bytes of the snap-in provided information (SPIN). The maximum length of the SPIN is 256 bytes.

Length of this entry. The length in bytes of this entry. This is used to get to the next entry.

Length of this message descriptor. The length in bytes of this message descriptor. The maximum length of a message descriptor is 16 million bytes.

Message descriptor data. One or more parameter list formats that follow the common header. Message descriptors are made up of a common header and a list of entries. The format of each entry in the list is defined by the format name associated with the message descriptor, which is located in the common header. The number of entries in the list is also defined in the common header.

Message type. The type of message that is associated with the entry.

Number of bytes available for this message descriptor. The number of bytes available in the space where the mail server framework puts the information being retrieved.

Number of entries available for this message descriptor. The total number of entries available for this message descriptor indicates the number of entries that would be returned if the length of the message descriptor is greater than or equal to the number of bytes available for this message descriptor.

Offset of the first entry in the message descriptor. The offset from the beginning of this message descriptor to the first entry in the list of entries.

Origin address. A string that represents the address associated with the originator of the message. The contents and format of the string are not defined by the mail server framework. It is assumed that the address type associated with the origin address defines the contents of the origin address field.

Origin address coded character set identifier (CCSID). The CCSID provided for the origin address. Valid values for the CCSID are 1 through 65533 and 65535.

Original recipient address. A string that represents the address associated with the original recipient of the message. The contents and format of the string are not defined by the mail server framework. It is assumed that the address type associated with the original recipient address defines the contents of the original recipient address field.

Original recipient address coded character set identifier (CCSID). The CCSID provided for the original recipient address. Valid values for the CCSID are 1 through 65533 and 65535.

Qualified mail server framework exit program name. The qualified program name of the program that was called at the mail server framework exit point.

Reason code. A code that identifies reasons associated with the message delivery to this address. In the case of a nondelivery entry, this field would contain the reason the delivery of this message to this recipient failed. The reason code is assumed to contain the X.400 nondelivery reason code.

Note: The reason code and diagnostic code fields are assumed to contain values that are documented in *CCITT Data Communication Networks Message Handling Systems 1988 Recommendation for X.400-X.420* for fields Non-delivery-reason-code and Non-delivery-diagnostic-code.

Receiver message descriptor pointer. A space pointer that points to a message descriptor. A message descriptor is used to pass information which describes the message.

Recipient address. A string that represents the address associated with a recipient of the message. The contents and format of the string are not defined by the mail server framework. It is assumed that the address type associated with the recipient address defines the contents of the recipient address field.

Recipient address coded character set identifier (CCSID). The CCSID provided for the recipient address. Valid values for the CCSID are 1 through 65533 and 65535.

Recipient status flag. A flag which when set to 1 indicates that this entry has been replaced by either one or multiple entries. Entries with this flag set to 1 are referred to as parents. Entries with this flag set zero are referred to as children.

Reply-to address. A string that represents the address to be replied to. The contents and format of the string are not defined by the mail server framework. It is assumed that the address type associated with the reply-to address defines the contents of the reply-to address field.

Reply-to address coded character set identifier (CCSID). The CCSID provided for the reply-to address. Valid values for the CCSID are 1 through 65533 and 65535.

Reply requested flag. Whether this original recipient should reply to the message. The possible values are as follows:

0	A reply is not requested from this original recipient
1	A reply is requested from this original recipient

Report-on address. A string which represents the address to be reported on. The contents and format of the string are not defined by the mail server framework. It is assumed that the address type associated with the report-on address defines the contents of the report-on address field.

Report-on address coded character set identifier (CCSID). The CCSID provided for the report-on address. Valid values for the CCSID are 1 through 65533 and 65535.

Report-to address. A string which represents the address to be reported to. The contents and format of the string are not defined by the mail server framework. It is assumed that the address type associated with the report-to address defines the contents of the report-to address field.

Report-to address coded character set identifier (CCSID). The CCSID provided for the report-to address. Valid values for the CCSID are 1 through 65533 and 65535.

Reserved. All reserved fields must be set to zero.

Return code from the exit program. The return code returned by the user exit program.

Snap-in provided information (SPIN). An area where snap-ins can store information that other snap-ins may use. SPIN provides a place where information relating to a specific recipient can be stored and used by snap-ins in the same or different exit points. This is completely user-defined and user-interpreted data.

Status. The status associated with each recipient entry. The possible values are:

1	Forwarded (remote)
2	Ignore
3	Local
4	Nondeliverable
5	Security violation

Timestamp of when the message was created. The timestamp of when the message was created. The format of the timestamp is CYYMMDDHHMMSSMMM. The fields are defined in the order that they occur in the string.

C	Century, where 0 indicates years 19*xx* and 1 indicates years 20*xx*.
YY	Year
MM	Month
DD	Day
HH	Hour
mm	Minute of the hour
SS	Second of the minute
MMM	Milliseconds in zoned format

Unique identifier. A unique identifier that differentiates each item within a particular list. Identifiers are generated for each list item when the Create Mail Message (QzmfCrtMailMsg) API has successfully completed. These unique identifiers are temporary and may change as the mail service processes a message. List entries are placed in message descriptors such that unique identifiers are in ascending order.

Unique identifier of parent entry. The unique identifier associated with the parent entry of this entry. A parent entry is an entry that had been replaced by a single or multiple entries.

Unique identifier of referenced entry. The unique identifier of another ORCL0100, ORGL0100, ENVL0100, RTAL0100, ROAL0100, RPYL0100, or ATTL0100 entry that this entry refers to. This field can be used to create entry cross-references as new entries are added.

Error Messages

None

Exception Messages That are Monitored By MSF

The following messages can be signaled by the snap-in program to the mail server framework.

Message ID	Exception Message Text
CPFAF90 E	End mail server framework (MSF) job.
CPFAF91 E	End processing MSF message.

Exit Program Introduced: V3R1

Track Mail Message Changes Exit Program

Required Parameter Group:

1	Mail message identifier	Input	Char(32)
2	Exit point name	Input	Char(20)
3	Qualified exit point program name	Input	Char(20)
4	Format name	Input	Char(8)
5	Return code	Output	Binary(4)

```
Exit Point Name:  QIBM_QZMFMSF_TRK_CHG
```

The exit point name and the name of an exit point program (which resulted in changes to a message) are passed to the Track Mail Message Changes exit program. The exit program configured for this exit is allowed to read all of the parts of a message to track the message. The exit point, however, cannot affect the processing of the message. The return codes are provided for tracking purposes and do not affect the processing of the message.

Required Parameter Group

Mail message identifier

INPUT; CHAR(32)

The variable containing the mail message identifier that the Track Mail Message Changes exit program is passing information for. The mail message identifier is composed of characters A through Z and 0 through 9 only.

Exit point name

INPUT; CHAR(20)

The name of the user exit point that was being processed when the changes to the message were made.

Qualified exit point program name

INPUT; CHAR(20)

The user qualified name of the program that was called that resulted in changes being made to the message.

Format name

INPUT; CHAR(8)

The variable containing the format name of the parameter list in which the Track Mail Message Changes exit program is passing information. This field must be set to TCMM0100.

Return code

OUTPUT; BINARY(4)

0	Everything was OK.

1	Data was not valid.
2	Severe error encountered.

Note: This return code is stored for tracking purposes and does not affect the processing of the message.

Exit Program Introduced: V3R1

Validate Data Field Exit Program

Required Parameter Group:

1	Mail message identifier	Input	Char(32)
2	Message descriptor attributes	Input	Array of Char(*)
3	Number of message descriptor attributes	Input	Binary(4)
4	Format name	Input	Char(8)
5	Return code	Output	Binary(4)

```
Exit Point Name:  QIBM_QZMFMSF_VLD_TYP
Exit Point Format Name:  VDFF0100
```

To configure the mail server framework, the user or programmer defines the format of information that it will use. The four basic type groups that the mail server framework supports are address types, message types, envelope types, and attachment types. The definition of a type consists of its name and its value. The mail server framework assumes that the definition of the type of the information defines the format of the information.

The mail server framework also provides a Validate Data Field exit program where a system can register programs to provide additional checking. These programs are called when mail messages are created or changed (using the Create Mail Message (QzmfCrtMailMsg) or Change Mail Message (QzmfChgMailMsg) API) and the message contains a type that has a validate data field exit program configured. The exit programs can be registered to be called for specific types of information only, or whenever any information for a particular type group is changed. Only list entries that have a matching type are passed to the user exit programs. The Validate Data Field exit program is called whenever any information matching the selection material is added to the message. When the program is complete, it returns its status in the return code parameter. If the return code from the exit program was zero, then the data is assumed to be valid and accepted. If the return code was not zero, then the data is assumed to be not valid and is rejected.

Required Parameter Group

Mail message identifier

INPUT; CHAR(32)

The variable containing the mail message identifier that the Validate Data Field exit program is passing information for. The mail message identifier is composed of characters A through Z and 0 through 9 only.

Message descriptor attributes

INPUT; ARRAY OF CHAR(*)

This array contains the message descriptor attributes for the data that is to be validated. The following table defines a message descriptor attributes array entry.

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	(PTR(SPP))	Message descriptor pointer

16	10	BINARY(4)	Length of the message descriptor
20	14	CHAR(8)	Message descriptor format name
28	1C	BINARY(4)	Reserved (must be set to zero)

Number of message descriptor attributes

OUTPUT; BINARY(4)

Tells the snap-in program how many message descriptor entries are in the array being passed.

Format name

INPUT; CHAR(8)

The variable containing the format name of the parameter list in which information is being passed. This field must be set to VDFF0100.

Return code

OUTPUT; BINARY(4)

0	Data was valid.
1	Data was not valid.
2	Severe error encountered

Message Descriptors

A message descriptor consists of:

- A Validate Data Field Exit common header section
- A list data section that contains the data to be validated. The following table shows valid format names for message descriptors that are passed to the validate data field exit program.

Format Name	**Description**
ORCL0100	Original recipient entry
ORGL0100	Originator entry
ENVL0100	Envelope entry
RCPL0100	Recipient entry
RPYL0100	Reply-to address entry
ROAL0100	Report-on address entry
RTAL0100	Report-to address entry
ATTL0100	Attachment reference entry

Validate Data Field Exit Common Header

The following table shows the common header section that is always included at the beginning of a message descriptor. It can be used with all of the formats that can be passed using this exit. There may be one or more entries after the common header section.

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	BINARY(4)	Length of this message descriptor
4	4	BINARY(4)	Reserved (must be set to zero)
8	8	CHAR(8)	Message descriptor format name
16	10	BINARY(4)	Offset of the first entry in the message descriptor
20	14	BINARY(4)	Number of entries in this message descriptor
24	18	BINARY(4)	Reserved (must be set to zero)
*	*	CHAR(*)	Message descriptor data

Message Descriptor Formats for Validate Data Field Exit

One of the following message descriptor formats is used to describe the data that follows the common header.

ORCL0100 Format (Original Recipient Entry)

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	BINARY(4)	Length of this entry
4	4	BINARY(4)	Displacement of the original recipient address from the beginning of this ORCL0100 entry
8	8	BINARY(4)	Length of the original recipient address
12	C	CHAR(4)	Address type
16	10	BINARY(4)	Original recipient address coded character set identifier(CCSID)
20	14	BINARY(4)	Distribution type
24	18	BINARY(4)	Reply requested flag
28	1C	BINARY(4)	Unique identifier
32	20	BINARY(4)	Unique identifier of referenced ORCL0100 entry
36	24	BINARY(4)	Reserved (must be set to zero)
*	*	CHAR(*)	Original recipient address

ORGL0100 Format (Originator Entry)

Offset			
Dec	Hex	Type	Field
0	0	BINARY(4)	Length of this entry
4	4	BINARY(4)	Displacement of the origin address from the beginning of this ORGL0100 entry
8	8	BINARY(4)	Length of origin address
12	C	CHAR(4)	Address type
16	10	BINARY(4)	Origin address coded character set identifier (CCSID)
20	14	BINARY(4)	Unique identifier
24	18	BINARY(4)	Unique identifier of referenced ORGL0100 entry
28	1C	BINARY(4)	Reserved (must be set to zero)
*	*	CHAR(*)	Origin address

ENVL0100 Format (Envelope Entry)

Offset			
Dec	Hex	Type	Field
0	0	BINARY(4)	Length of this entry
4	4	BINARY(4)	Displacement of envelope from the beginning of this ENVL0100 entry
8	8	BINARY(4)	Length of envelope
12	C	CHAR(4)	Envelope type
16	10	BINARY(4)	Unique identifier
20	14	BINARY(4)	Unique identifier of referenced ENVL0100 entry
24	18	BINARY(4)	Reserved (must be set to zero)
*	*	CHAR(*)	Envelope

RCPL0100 Format (Recipient Entry)

Offset			
Dec	Hex	Type	Field
0	0	BINARY(4)	Length of this entry
4	4	BINARY(4)	Displacement of the snap-in provided information (SPIN) from the beginning of this RCPL0100 entry
8	8	BINARY(4)	Length of snap-in provided information (SPIN)

12	C	BINARY(4)	Displacement of the recipient address from the beginning of this RCPL0100 entry
16	10	BINARY(4)	Length of recipient address
20	14	CHAR(4)	Address type
24	18	BINARY(4)	Recipient address coded character set identifier (CCSID)
28	1C	BINARY(4)	Reason code
32	20	BINARY(4)	Diagnostic code
36	24	CHAR(4)	Message type
40	28	BINARY(4)	Status
44	2C	BINARY(4)	Reserved
48	30	BINARY(4)	Unique identifier
52	34	BINARY(4)	Reserved (must be set to zero)
*	*	CHAR(*)	Recipient address
*	*	CHAR(*)	Snap-in provided information (SPIN)

ROAL0100 Format (Report-on Address Entry)

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	BINARY(4)	Length of this entry
4	4	BINARY(4)	Displacement of the snap-in provided information (SPIN) from the beginning of this ROAL0100 entry
8	8	BINARY(4)	Length of snap-in provided information (SPIN)
12	C	BINARY(4)	Displacement of the report-on address from the beginning of this ROAL0100 entry
16	10	BINARY(4)	Length of address
20	14	CHAR(4)	Address type
24	18	BINARY(4)	Report-on address coded character set identifier (CCSID)
28	1C	BINARY(4)	Reason code
32	20	BINARY(4)	Diagnostic code
36	24	BINARY(4)	Unique identifier
40	28	BINARY(4)	Unique identifier of referenced ROAL0100 entry
44	2C	BINARY(4)	Reserved
*	*	CHAR(*)	Report-on address
*	*	CHAR(*)	Snap-in provided information (SPIN)

RPYL0100 Format (Reply-to Address Entry)

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Length of this entry
4	4	BINARY(4)	Displacement of the reply-to address from the beginning of this RPYL0100 entry
8	8	BINARY(4)	Length of reply-to address
12	C	CHAR(4)	Address type
16	10	BINARY(4)	Reply-to address coded character set identifier (CCSID)
20	14	BINARY(4)	Unique identifier
24	18	BINARY(4)	Unique identifier of referenced RPYL0100 entry
28	1C	BINARY(4)	Reserved (must be set to zero)
*	*	CHAR(*)	Reply-to address

RTAL0100 Format (Report-to Address Entry)

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Length of this list entry
4	4	BINARY(4)	Displacement of the report-to address from the beginning of this RTAL0100 entry
8	8	BINARY(4)	Length of address
12	C	CHAR(4)	Address type
16	10	BINARY(4)	Report-to address coded character set identifier (CCSID)
20	14	BINARY(4)	Unique identifier
24	18	BINARY(4)	Unique identifier of referenced RTAL0100 entry
28	1C	BINARY(4)	Reserved (must be set to zero)
*	*	CHAR(*)	Report-to address

ATTL0100 Format (Attachment Reference Entry)

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Length of this list entry
4	4	BINARY(4)	Displacement of the attachment reference from the beginning of this ATTL0100 entry

8	8	BINARY(4)	Length of attachment reference
12	C	CHAR(4)	Attachment reference type
16	10	BINARY(4)	Unique identifier
20	14	BINARY(4)	Unique identifier of referenced ATTL0100 entry
24	18	BINARY(4)	Reserved (must be set to zero)
*	*	CHAR(*)	Attachment reference

Field Descriptions

Address type. The type of address that is contained in the entry.

Attachment reference. A reference to an attachment that is associated with the message.

Attachment reference type. The type of attachment that is contained in the entry.

Diagnostic code. A code that indicates the cause of a problem which caused the delivery of a message to this address to fail. The diagnostic code is assumed to contain the X.400 nondelivery diagnostic code.

Note: The reason code and diagnostic code fields are assumed to contain values that are documented in *CCITT Data Communication Networks Message Handling Systems 1988 Recommendation for X.400-X.420* for fields Non-delivery-reason-code and Non-delivery-diagnostic-code.

Displacement of the attachment reference from the beginning of this ATTL0100 entry. The displacement from the beginning of this entry to the attachment reference in this entry.

Displacement of the envelope from the beginning of this ENVL0100 entry. The displacement from the beginning of this entry to the envelope in this entry.

Displacement of the origin address from the beginning of this ORGL0100 entry. The displacement from the beginning of this entry to the origin address in this entry.

Displacement of the original recipient address from the beginning of this ORCL0100 entry. The displacement from the beginning of this entry to the original recipient entry in this entry.

Displacement of the recipient address from the beginning of this RCPL0100 entry. The displacement from the beginning of this entry to the recipient address in this entry.

Displacement of the reply-to address from the beginning of this RPYL0100 entry. The displacement from the beginning of this entry to the reply-to address in this entry.

Displacement of the report-to address from the beginning of this RTAL0100 entry. The displacement from the beginning of this entry to the report-to address in this entry.

Displacement of the report-on address from the beginning of this ROAL0100 entry. The displacement from the beginning of this entry to the report-on address in this entry.

Distribution type. The type of distribution associated with each recipient entry. The possible values are:

0	Normal message recipient
1	CC (carbon copy) recipient
2	BCC (blind carbon copy) recipient

Envelope. A string of data representing information about the message, aside from the attachments and its recipients.

Envelope type. The type of envelope that is contained in the entry.

Exit program number. A number assigned to the exit program when it is registered using the registration facility. This is the number in effect at the time the exit program was called.

Format name. The content and format of the information provided for each message parameter. The possible values are:

ORCL0100	Original recipient entry
ORGL0100	Originator entry
ENVL0100	Envelope entry
RCPL0100	Recipient entry
ROAL0100	Report-on address entry
RPYL0100	Reply-to address entry
RTAL0100	Report-to address entry
ATTL0100	Attachment reference entry

Length of address. The length of address that is contained in the entry. The maximum length of an address is 1024 bytes.

Length of attachment reference. The length in bytes of the attachment reference that is contained in the entry.

Length of envelope. The length of envelope that is contained in the entry.

Length of origin address. The length of the origin address for this entry. The maximum length of an origin address is 1024 bytes.

Length of original recipient address. The length of the original recipient address for this entry. The maximum length of an original recipient address is 1024 bytes.

Length of receiver message descriptor. The length in bytes of the message descriptor that is being pointed to by the pointer to the message descriptor. The maximum length of a message descriptor is 16 million bytes.

Length of recipient address. The length in bytes of the recipient address. The maximum length of a recipient address is 1024 bytes.

Length of reply-to address. The length in bytes of the reply-to address. The maximum length of a reply-to address is 1024 bytes.

Length of snap-in provided information (SPIN). The length in bytes of the snap-in provided information (SPIN). The maximum length of the SPIN is 256 bytes.

Length of this entry. The length in bytes of this entry. This is used to get to the next entry.

Length of this message descriptor. The length in bytes of this message descriptor. The maximum length of a message descriptor is 16 million bytes.

Message descriptor data. One or more parameter list formats that follow the common header. Message descriptors are made up of a common header and a list of entries. The format of each entry in the list is defined by the format name associated with the message descriptor, which is located in the common header. The number of entries in the list is also defined in the common header.

Message type. The type of message that is associated with the entry.

Number of bytes available for this message descriptor. The number of bytes available in the space where the mail server framework puts the information being retrieved.

Number of entries available for this message descriptor. The total number of entries available for this message descriptor indicates the number of entries that would be returned if the length of the message descriptor is greater than or equal to the number of bytes available for this message descriptor.

Offset of the first entry in the message descriptor. The offset from the beginning of this message descriptor to the first entry in the list of entries.

Origin address. A string that represents the address associated with the originator of the message. The contents and format of the string are not defined by the mail server framework. It is assumed that the address type associated with the origin address defines the contents of the origin address field.

Origin address coded character set identifier (CCSID). The CCSID provided for the origin address. Valid values for the CCSID are 1 through 65533 and 65535.

Original recipient address. A string that represents the address associated with the original recipient of the message. The contents and format of the string are not defined by the mail server framework. It is assumed that the address type associated with the original recipient address defines the contents of the original recipient address field.

Original recipient address coded character set identifier (CCSID). The CCSID provided for the original recipient address. Valid values for the CCSID are 1 through 65533 and 65535.

Reason code. A code that identifies reasons associated with the message delivery to this address. In the case of a nondelivery entry, this field would contain the reason the delivery of this message to this recipient failed. The reason code is assumed to contain the X.400 nondelivery reason code.

Note: The reason code and diagnostic code fields are assumed to contain values that are documented in *CCITT Data Communication Networks Message Handling Systems 1988 Recommendation for X.400-X.420* for fields Non-delivery-reason-code and Non-delivery-diagnostic-code.

Recipient address. A string that represents the address associated with a recipient of the message. The contents and format of the string are not defined by the mail server framework. It is assumed that the address type associated with the recipient address defines the contents of the recipient address field.

Recipient address coded character set identifier (CCSID). The CCSID provided for the recipient address. Valid values for the CCSID are 1 through 65533 and 65535.

Recipient status flag. A flag which when set to 1 indicates that this entry has been replaced by either one or multiple entries. Entries with this flag set to 1 are referred to as parents. Entries with this flag set zero are referred to as children.

Reply requested flag. Whether this original recipient should reply to the message. The possible values are:

0	A reply is not requested from this original recipient
1	A reply is requested from this original recipient

Reply-to address. A string that represents the address to be replied to. The contents and format of the string are not defined by the mail server framework. It is assumed that the address type associated with the reply-to address defines the contents of the reply-to address field.

Reply-to address coded character set identifier (CCSID). The CCSID provided for the reply-to address. Valid values for the CCSID are 1 through 65533 and 65535.

Report-on address. A string which represents the address to be reported on. The contents and format of the string are not defined by the mail server framework. It is assumed that the address type associated with the report-on address defines the contents of the report-on address field.

Report-on address coded character set identifier (CCSID). The CCSID provided for the report-on address. Valid values for the CCSID are 1 through 65533 and 65535.

Report-to address. A string which represents the address to be reported to. The contents and format of the string are not defined by the mail server framework. It is assumed that the address type associated with the report-to address defines the contents of the report-to address field.

Report-to address coded character set identifier (CCSID). The CCSID provided for the report-to address. Valid values for the CCSID are 1 through 65533 and 65535.

Reserved. All reserved fields must be set to zero.

Return code from the exit program. The return code returned by the user exit program.

Snap-in provided information (SPIN). An area where snap-ins can store information that other snap-ins may use. SPIN provides a place where information relating to a specific recipient can be stored and used by snap-ins in the same or different exit points. This is completely user-defined and user-interpreted data.

Status. The status associated with each recipient entry. The possible values are:

1	Forwarded (remote)
2	Ignore
3	Local
4	Nondeliverable
5	Security violation

Unique identifier. A unique identifier that differentiates each item within a particular list. Identifiers are generated for each list item when the Create Mail Message (QzmfCrtMailMsg) API has successfully completed. These unique identifiers are temporary and may change as the mail service processes a message. List entries are placed in message descriptors such that unique identifiers are in ascending order.

Unique identifier of referenced entry. The unique identifier of another ORCL0100, ORGL0100, ENVL0100, RTAL0100, ROAL0100, RPYL0100, or ATTL0100 entry that this entry refers to. This field can be used to create entry cross-references as new entries are added.

Exit Program Introduced: V3R1

SNADS File Server Object APIs

The SNADS file server object APIs would be used if you were dealing with AnyMail/400 mail server framework (MSF) messages that contained SNADS attachments. You would use the file server object read API if the MSF message was originated by OfficeVision(R) or object distribution. You would use the create and write APIs if the MSF message was destined for an OfficeVision or an object distribution user, or if you want to temporarily store attachments.

With the SNADS file server interface, you can create and manipulate either SNADS general file server objects or Document Interchange Architecture (DIA) file server objects. A file server object (FSO) in this context is used by SNADS and by applications like object distribution and OfficeVision to store data associated with a message (for example, an OfficeVision note).

Although SNADS general file server objects and DIA file server objects are manipulated using the same interface, they differ in content.

For additional information, see File Server Objects.

The SNADS file server object APIs are:

- Assign SNADS File Server Object Access ID (QZDASNID) assigns an access ID to a specific file server object (FSO).
- Create SNADS File Server Object (QZDCRFSO) allows a user to create an FSO on the server.
- List SNADS File Server Object Access IDs (QZDLSTID) lists the current accesses owned by the specified product.
- Read SNADS File Server Object (QZDRDFSO) provides functions for reading a file server object on the server.
- Retrieve SNADS File Server Object Access ID (QZDRTVID) allows the caller to retrieve information about one access ID.
- Revoke SNADS File Server Object Access ID (QZDRVKID) revokes an access ID to a specific file server object that was previously assigned.
- Write to SNADS File Server Object (QZDWTFSO) provides functions for writing to a file server object on the system.

Top | Office APIs | APIs by category

File Server Objects

There are OS/400 system considerations, actions, or functions that might be performed that could affect file server objects.

Following are some system considerations related to how the internal file server objects are used by the file server object APIs.

- All file server objects created using the FSO APIs are stored in the QUSRSYS library. If that library is ever cleared, all file server objects are deleted.
- All file server objects are permanent system objects. They are not deleted due to a system IPL (normal or abnormal) or due to a normal installation of OS/400.
- All file server objects are owned by the QSNADS user profile. To determine the amount of space used by the system for file server objects, you can display the amount of storage owned by the QSNADS user profile.
- To manage the existence of a file server object, you would use the assign and revoke access ID APIs. However, if these APIs are not used, there is no protection on the file server object (see Assign and Revoke APIs Example). If someone issues the Reclaim Storage (RCLSTG) command and there is no protection on a file server object, that file server object will be deleted.
- File server objects cannot be saved. When you perform a save of the system, file server objects currently existing on that system are not saved. File server objects that were created on one system cannot be restored on another system.

File Servers Supported by FSO APIs

The following sections talk about the two file servers that are supported by the FSO APIs. These file servers are programs that are called by the APIs to perform specific operations. Both file servers have the same interface. To indicate which file server program should be used by the API, you pass in the name of the file server on the API call.

Note: Only one file server can be used for any one file server object. For example, to manipulate file server object A, you cannot use the DIA file server to create the object and then the SNADS general file server to write data to the object. Only one of the file servers may be used to do both operations. Likewise, if file server object A was created using the DIA file server, the DIA file server must be used to read the object later.

SNADS General File Server

The SNADS general file server is used to create objects (called SNADS general file server objects (FSOs)) which are simply a stream of bytes. When this type of FSO is created, no data conversions take place and the only information in the FSO is the data itself. For example, the SNADS general file server is used by object distribution to store data for files or spooled files.

DIA File Server

The DIA file server is used to create DIA FSOs. These objects are used by OfficeVision and can consist of more information than just the data. Examples include the subject, origination date, expiration date, and priority. This additional information is stored in the IDP (interchange document profile) in the document that comes right before the actual document data.

The data in a DIA document are the actual contents of an OfficeVision note, message or document. The data in a DIA FSO can take on several different formats that include final format text (FFT), revised format text (RFT), or PC file format. See the Document Interchange Architecture: Technical Reference, C23-0781-01, for more details on DIA.

Assign and Revoke APIs Example

Within SNADS there are several jobs that work with an MSF message. When an MSF message has a file server object attached to it, each job must secure its usage of the FSO by assigning an access ID to that FSO.

1. When the first job has assigned an access ID and completed its work on the MSF message, it passes the message to the next job.

2. The next job assigns an access ID for itself, but also revokes the access ID of the previous job.

3. The last job to work with the MSF message never actually assigns an access ID to the FSO, but it does revoke the access ID of the previous job.

When all access IDs to an FSO have been revoked, the file server object is deleted.

How Usage Counts Are Used

The SNADS general file server uses a usage count to keep track of the use of a file server object. The usage count indicates how many MSF messages are referencing the object. For example, as an MSF message (that refers to an attachment) flows through SNADS, the MSF message is processed by several different jobs. These jobs may each make copies of the MSF message to change some of the distribution data. The attachment, or file server object, could then be referred to by more than one MSF message at a time. For each reference, the Assign SNADS File Server Object Access ID (QZDASNID) API should be called to increment the usage count by one. As each job finishes its processing of the distribution, it calls the Revoke SNADS File Server Object Access ID (QZDRVKID) API. This decrements the usage count by one. When the usage count becomes zero, the file server object is destroyed.

The assigning and revoking operations for DIA documents follows the same methodology as the SNADS general file server. The usage count for DIA documents is stored in the distribution tracking object (DTO) for the document. After the usage count has been decremented by calling the QZDRVKID API, the document and DTO are deleted if they are eligible for deletion.

Assign SNADS File Server Object Access ID (QZDASNID) API

Required Parameter Group:

1	File server object handle	Input	Char(32)
2	Correlation	Input	Char(3000)
3	Correlation bytes provided	Input	Binary(4)
4	Product ID	Input	Char(7)
5	Access ID	Output	Char(8)
6	Error code	I/O	Char(*)

Threadsafe: No

The Assign SNADS File Server Access ID (QZDASNID) API assigns an access ID to a specific file server object (FSO). An access ID lets the system know that a product is currently working with the file server object and that the file server object should not be deleted. An access ID is returned to the caller so that the access can be revoked at a later time. This API also increments the usage count of the FSO by one. Access IDs can be assigned to a particular file server object, and multiple accesses can be assigned at the same time. You can assign an access ID to an FSO 1 through 2 147 483 647 times, which equates to the usage count.

Authorities and Locks

API Public Authority

*EXCLUDE

Required Parameter Group

File server object handle

INPUT; CHAR(32)

The necessary linkage to the file server object. The value of this parameter should be taken from the FSO handle that was returned on the call to the Create SNADS File Server Object (QZDCRFSO) API.

Correlation

INPUT; CHAR(3000)

Information defined by the caller that identifies the file server object or the file server object access ID. For example, the correlation could contain the message identifier of the message attached to the file server object. If the correlation bytes provided field is zero, this indicates that no correlation is to be stored with the FSO.

Correlation bytes provided

INPUT; BINARY(4)

This field specifies the length of the correlation data.

Product ID

INPUT; CHAR(7)

The identifier of the product that is using the QZDASNID API. The product ID is used together with the access ID to identify the file server object access.

Valid values for this parameter are as follows:

Product ID	The product ID of a product that is installed on the iSeries server.
QMSFPRD	The MSF product identifier. This value can be used for all file server objects that are referred to by the MSF messages.

Access ID

OUTPUT; CHAR(8)

The identifier used together with the product ID to identify the file server object access.

Error code

I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3A0D E	Temporary server error.
CPF3A09 E	System error.
CPF3A12 E	Interface error.
CPF3C90 E	Literal value cannot be changed.
CPF3CF1 E	Error code parameter not valid.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API Introduced: V3R6

Top | Office APIs | APIs by category

Create SNADS File Server Object (QZDCRFSO) API

Required Parameter Group:

1	File server object handle	Output	Char(32)
2	File server object data length	Input	Binary(4)
3	File server name structure	Input	Char(68)
4	Error code	I/O	Char(*)

Threadsafe: No

The Create SNADS File Server Object (QZDCRFSO) API allows a user to create a file server object (FSO). A file server object also may be referred to as an attachment (MSF messages, for example, can have attachments). The types of file server objects that the QZDCRFSO API builds are SNADS general file-server objects and Document Interchange Architecture (DIA) file-server objects. The interface is the same to these two types of objects.

Authorities and Locks

API Public Authority

*EXCLUDE

Required Parameter Group

File server object handle

OUTPUT; CHAR(32)

The necessary linkage to the file server object. This parameter is set as output on the create operation and is used as input on all other file server functions.

File server object data length

INPUT; BINARY(4)

The calculated length of the data to be written. The API creates an FSO space (up to 16MB) that is big enough to hold the length of data specified. If you specify too small a size, there is no problem. SNADS creates an FSO of the size you specify, but if the actual size is larger, SNADS extends the FSO. For better performance, you should specify the actual size of the FSO. Valid values for this parameter are:

1 byte through 16 megabytes

File server name structure

INPUT; CHAR(68)

The file server name (either DIA or SNADS) and the server name length.

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Length of file server name
4	4	CHAR(64)	File server name

The length of the file server name is the number of bytes in the file server name defined by the SNADS architecture. Valid values for length of the file server name are as follows:

4	SNADS general file-server name
4	DIA file-server name

The file server name is the name value defined by the SNADS architecture for the file server that is to be used. The only file servers available with this API are the SNADS general file server and the DIA file server. Valid values for file server name:

'21F0F0F6'X	SNADS general file-server name
'20F0F0F1'X	DIA file-server name

Error code

I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3A0D E	Temporary server error.
CPF3A09 E	System error.
CPF3A1D E	Limit of number of open file server objects exceeded.
CPF3A17 E	Permanent server error.
CPF3C90 E	Literal value cannot be changed.
CPF3CF1 E	Error code parameter not valid.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API Introduced: V3R6

List SNADS File Server Object Access IDs (QZDLSTID) API

Required Parameter Group:

1	Qualified user space name	Input	Char(20)
2	Format name	Input	Char(8)
3	Product ID	Input	Char(7)
4	Continuation access ID	Input	Char(8)
5	Error code	I/O	Char(*)

Threadsafe: No

The List SNADS File Server Object Access IDs (QZDLSTID) API can be used to list the current accesses owned by the specified product. Each access ID represents an access done on a file server object (FSO) that was obtained by using the Assign SNADS File Server Object Access ID (QZDASNID) API. The list of access IDs is returned in a user space.

This API provides a way for an application to do cleanup. For example, if the assigned access IDs that are stored are lost, you can use this API to find out what access IDs need to be revoked. An application should use this list interface at some periodic interval to check if it has any assigned access IDs that were not revoked.

If you use the MSF product ID (QMSFPRD) on the QZDASNID API, you do not have to use this API for cleanup purposes. Cleanup routines currently exist in the system to revoke access IDs assigned to the MSF product. These cleanup routines revoke access IDs only when the IDs are no longer associated with any file server object.

The continuation access ID parameter should be either set to nulls (X'00') to have this API start the search at the beginning of all access IDs, or set to a specific access ID. For the latter, the QZDLSTID API starts searching with the next greater access ID. (No error is returned if the starting access ID does not exist.) If more access IDs exist for a product than were returned on the call to the QZDLSTID API, this API sets the information status variable in the generic user space header to indicate partially complete information. To retrieve more access IDs, the caller should use the continuation access ID as input for the next call to the QZDLSTID API.

Authorities and Locks

API Public Authority

*EXCLUDE

Required Parameter Group

User space name

INPUT; CHAR(20)

The user space that stores all the access IDs found. This structure contains the user space name and the library name where the user space is defined. The first 10 characters contain the user space name, and the second 10 characters contain the library name.

Format name

INPUT; CHAR(8)

The name of the format that returns access ID information. You can specify this format:

ACID0100	Each entry contains an access ID.

Product ID

INPUT; CHAR(7)

The identifier of the product that is using the QZDLSTID API. This parameter is necessary to group access IDs by product and to facilitate better cleanup of file server objects. For example, an installed product that was creating FSOs is removed from the system, but FSOs still exist that are referred to by that product. The SNADS cleanup routines check to ensure that the product is installed, and if the product was removed, the SNADS cleanup routines delete the remaining FSOs referred to by that product.

Valid values for this parameter are as follows:

Product ID	The product ID of a product that is installed on the iSeries server.
QMSFPRD	The MSF product identifier. This value can be used for all file server objects that are referred to by the MSF messages.

Continuation access ID

INPUT; CHAR(8)

The identifier that is used together with the product ID to identify the first file server object access to find. You can determine if a previous call resulted in partially complete information by checking the information status variable in the generic user-space header following the API call.

If the API is not attempting to continue from a previous call, this parameter must be set to hexadecimal zeros. Otherwise, a valid continuation value must be supplied. The value may be obtained from the list header section of the user space used in the previous call. When continuing, the first entry in the returned list is the entry that immediately follows the last entry returned in the previous call.

Error code

I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter.

Format of the User Space Variables

The following tables describe the order and format of the data returned in the user space.

Input Parameter Section

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	CHAR(10)	User space name specified
10	A	CHAR(10)	User space library name specified
20	14	CHAR(8)	Format name
28	1C	CHAR(7)	Product ID
35	23	CHAR(8)	Continuation access ID

Header Section

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	CHAR(8)	Continuation access ID

ACID0100 Format

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	CHAR(8)	Access ID

Field Descriptions

Access ID. Each entry in the returned list contains an access ID that is currently assigned to the product specified on input.

Continuation access ID (header section). A continuation point for the API. This value is set based on the contents of the information status variable in the generic header for the user space. The following situations can occur for the following information statuses:

C	The information returned in the user space is valid and complete. No continuation is necessary and the continuation access ID is set to nulls (X'00').
P	The information returned in the user space is valid but incomplete. The user may call the API again, starting where the last call left off. The continuation access ID contains a value, which may be supplied as an input parameter in later calls.
I	The information returned in the user space is not valid and not complete. The content of the continuation access ID is unpredictable.

Continuation access ID (input section). The identifier that is used to continue from a previous call to this API, which resulted in partially complete information.

Format name. The name of the format used to return access ID information.

Product ID. The identifier of the product that is using the QZDLSTID API.

User space library name specified. The name of the library that contains the user space.

User space name specified. The name of the user space to which the list of access IDs is returned.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3A09 E	System error.
CPF3A12 E	Interface error.
CPF3A13 E	Access ID not valid.
CPF3C90 E	Literal value cannot be changed.
CPF3CF1 E	Error code parameter not valid.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API Introduced: V3R6

Top | Office APIs | APIs by category

Read SNADS File Server Object (QZDRDFSO) API

Required Parameter Group:

1	File server object handle	Input	Char(32)
2	Operation requested	Input	Binary(4)
3	File server object data length	Output	Binary(4)
4	File server name structure	Input	Char(68)
5	Resume position	Input	Binary(4)
6	Buffer area	Input	Char(*)
7	Bytes provided in buffer area	Input	Binary(4)
8	Bytes available in buffer area	Output	Binary(4)
9	Error code	I/O	Char(*)

Threadsafe: No

The Read SNADS File Server Object (QZDRDFSO) API has four operations that can be used to read an FSO. The three required operations are **initiate read**, **read**, and **terminate read**. The initiate read operation opens up an existing file server object that is based on the information given to it by the caller. Once this operation is called, the read operation can be used to read the data contained in the file server object. The read operation can be called multiple times to read all the data. After all read operations are completed, the terminate read operation must be issued to close the file server object.

The optional operation provided on the QZDRDFSO API is **initiate read location**. This operation can be used to start reading the file server object at a specific location.

The QZDRDFSO API operates on SNADS general file server objects and DIA file server objects. The interface to both of these types of objects is essentially the same.

SNADS General File Server

The SNADS general file server provides a means for SNADS support to save the data object portion of a distribution request. The general file server stores objects in a byte stream format. No transformations are done to the data when it is written into or read out of a general file server object. The SNADS general file server objects are owned by QSNADS and are stored in the QUSRSYS library.

The SNADS general file server can support objects up to 4 gigabytes. Each file server object space can be only 16MB large, but the SNADS general file server has the capability of chaining file server objects together. If the amount of data is greater than 16MB, the SNADS general file server creates as many file server object spaces as necessary and then chains them together. This function is transparent to the caller of the SNADS general file server routines.

The SNADS general file server read operations are as follows:

Initiate Read

The initiate read operation does the setup necessary to read the general file server object. This includes finding the object specified by the object handle in the input parameters. This operation must be the first called before any of the other read operations can be used for the specific file server object. The initiate read operation only needs to be called once in the sequence of read operations.

Read

The caller of the read operation must specify what object is to be read, where to put the data that is read, and how much data is to be read. If the amount of actual data is less than the size of the caller's buffer space (that is, where the data will be read in), the read operation reads as much data as it can and then returns. Because a space can be a maximum of 16MB, the caller's buffer space should be no more than 16MB, and the caller should not request more data than 16MB to be read. If the object being read is larger than 16MB, the caller must call the read operation more than once to read all the data. If more than one read is done, the SNADS general file server read operation keeps a marker in the data (offset inside the object) so that on subsequent read operations, the read operation knows where to start reading.

Terminate Read

The terminate read operation does the cleanup processing after you have completed reading the data in the file server object. After this operation is called, the file server object may still exist, but an initiate read operation must be done to read the file server object again.

Initiate Read Location

The initiate read location operation is similar to the initiate read operation, but allows a caller to set up a general file server object to begin reading at a specified location rather than at the beginning of the object. This operation also needs to be called only once in the sequence of read operations. The caller of this operation must pass in the resume position of where to continue reading the file server object. This number is based on how many bytes were read when the terminate read operation was called.

DIA File Server

The DIA file server provides a means for SNADS support to save the data object portion of an OfficeVision distribution request. The DIA file server objects are of the type *DTO (distribution tracking object), are owned by the QDOC user profile, and are stored in the QUSRSYS library.

The DIA file server can support objects up to 2 gigabytes. Each file server object space can be only 16MB large, but the DIA file server has the capability of chaining file server objects together. If the amount of data is greater than 16MB, the DIA file server creates as many file server object spaces as necessary and then chains them together. This function is transparent to the caller of the DIA file server routines.

The DIA file server read operations are as follows:

Initiate Read

The initiate read operation does the setup necessary to read the DIA distribution tracking object. This includes finding the object specified by the object handle in the input parameters. This operation must be the first operation called before any of the other read operations can be used for the specific file server object.

Read

The caller of the read operation must specify what document object is to be read, where to put the data that is read, and how much data is to be read. If the amount of actual data is less than the size of the caller's buffer space (that is, where the data will be read in), the read operation reads as much data as it can and then returns.

Because a space can be a maximum of 16MB, the caller's buffer space should be no more than 16MB, and the caller should not request more data than 16MB to be read. If the document object being read is larger than

16MB, the caller must call the read operation more than once to read all the data. If more than one read operation is done, the DIA file server read operation keeps a marker in the data (offset inside the object) so that on subsequent read operations, the read operation knows where to start reading.

Terminate Read

The terminate read operation should be called when you have completed reading the document. This operation closes the document object.

Authorities and Locks

API Public Authority

*EXCLUDE

Required Parameter Group

File server object handle

INPUT; CHAR(32)

The necessary linkage to the file server object is contained in this parameter. This parameter must be specified on all file server read operations. The value of this parameter should be taken from the file server object handle that was returned on the call to the Create SNADS File Server Object (QZDCRFSO) API.

Operation requested

INPUT; BINARY(4)

The operation the caller is requesting. Valid values for this parameter are as follows:

Initiate read	1
Read	2
Terminate read	3
Initiate read location	9

File server object data length

OUTPUT; BINARY(4)

The total length of the file server object being read. This parameter is returned on the initiate read and the initiate read location functions.

File server name structure

INPUT; CHAR(68)

The file server name (either DIA or SNADS) and the server name length.

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	BINARY(4)	Length of file server name

4	4	CHAR(64)	File server name

The length of the file server name is the number of bytes in the file server name defined by the SNADS architecture. Valid values for length of the file server name are as follows:

4	SNADS general file-server name
4	DIA file-server name

The file server name is the name value defined by the SNADS architecture for the file server that is to be used. The only file servers available with this API are the SNADS general file server and the DIA file server. Valid values for file server name:

'21F0F0F6'X	SNADS general file-server name
'20F0F0F1'X	DIA file-server name

Resume position

INPUT; BINARY(4)

The place where the initiate read location operation is to resume reading in the FSO. This parameter is ignored on the initiate read, read, and terminate read operations. Valid values for this parameter follow:

1 through the current file server object data length

Buffer area

INPUT; CHAR(*)

A buffer area defined by the caller. It is used by the read operation to store the data that is being read.

Bytes provided in buffer area

INPUT; BINARY(4)

The number of bytes of data that can be read into the buffer area. Valid values for this parameter are as follows:

1 byte through 16 megabytes

Bytes available in buffer area

OUTPUT; BINARY(4)

The number of actual bytes of data that was read into the buffer area.

Error code

I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3A09 E	System error.
CPF3A1A E	Location in file server object not valid.
CPF3A1B E	File server object operation not valid.
CPF3A1C E	File server object request not valid.
CPF3A1D E	Limit of number of open file server objects exceeded.
CPF3A1E E	End of data reached in file server object.
CPF3A15 E	File server object not found.
CPF3A17 E	Permanent server error.
CPF3C90 E	Literal value cannot be changed.
CPF3CF1 E	Error code parameter not valid.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API Introduced: V3R6

Retrieve SNADS File Server Object Access ID (QZDRTVID) API

Required Parameter Group:

1	Product ID	Input	Char(7)
2	Access ID	Input	Char(8)
3	File server object handle	Output	Char(32)
4	Correlation	Output	Char(3000)
5	Correlation bytes available	Output	Binary(4)
6	Error code	I/O	Char(*)

Threadsafe: No

The Retrieve SNADS File Server Object Access ID (QZDRTVID) API allows the caller to retrieve information about one access ID. This information includes the file server object handle for the access ID and any correlation information associated with the access ID.

Authorities and Locks

API Public Authority

*EXCLUDE

Required Parameter Group

Product ID

INPUT; CHAR(7)

The identifier of the product that is using the List SNADS File Server Object Access IDs (QZDLSTID) API. This parameter is necessary to group access IDs by product and to facilitate better cleanup of file server objects. For example, an installed product that was creating FSOs is removed from the system, but FSOs still exist that are referred to by that product. The SNADS cleanup routines check to ensure that the product is installed, and if the product was removed, the SNADS cleanup routines delete the remaining FSOs referred to by that product.

Valid values for this parameter are as follows:

The product ID of a product that is installed on the iSeries server.

QMSFPRD	MSF product identifier. This value can be used for all file server objects that are referred to by the MSF messages.

Access ID

INPUT; CHAR(8)

The identifier that is used together with the product ID to identify the file server object access to find. Valid values for this parameter are as follows:

'0000000000000000'X through 'FFFFFFFFFFFFFFFF'X

File server object handle

OUTPUT; CHAR(32)

The necessary linkage to the file server object that was found by this API is contained in this parameter.

Correlation

OUTPUT; CHAR(3000)

The correlation is further identification of a file server object access that was found by this API. The correlation information is defined by the caller.

Correlation bytes available

OUTPUT; BINARY(4)

The number of bytes of correlation data returned. Valid values for this parameter are:

0 through 3000

Error code

I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3A09 E	System error.
CPF3A12 E	Interface error.
CPF3A13 E	Access ID not valid.
CPF3C90 E	Literal value cannot be changed.
CPF3CF1 E	Error code parameter not valid.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API Introduced: V3R6

Revoke SNADS File Server Object Access ID (QZDRVKID) API

Required Parameter Group:

1	Product ID	Input	Char(7)
2	Access ID	Input	Char(8)
3	Error code	I/O	Char(*)

Threadsafe: No

The Revoke SNADS File Server Object Access ID (QZDRVKID) API revokes an access ID to a specific file server object (FSO) that was previously assigned. Revoking an access lets the system know that the current process has finished working with the file server object. The access ID that was originally assigned to the file server object must be passed as input. This API also decrements the usage count of the file server object by one.

Authorities and Locks

API Public Authority

*EXCLUDE

Required Parameter Group

Product ID

INPUT; CHAR(7)

The identifier of the product that is using the QZDLSTID API. This parameter is necessary to group access IDs by product and to facilitate better cleanup of file server objects. For example, an installed product that was creating FSOs is removed from the system, but FSOs still exist that are referred to by that product. The SNADS cleanup routines check to ensure that the product is installed, and if the product was removed, the SNADS cleanup routines delete the remaining FSOs referred to by that product.

Valid values for this parameter are as follows:

Product ID	The product ID of a product that is installed on the iSeries server.
QMSFPRD	The MSF product identifier. This value can be used for all file server objects that are referred to by the MSF messages.

Access ID

INPUT; CHAR(8)

The identifier that is used along with the product ID to identify the file server object access. The value of this parameter should be taken from the access ID that was returned on the call to the Assign SNADS File Server Object Access ID (QZDASNID) API.

Error code

I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3A09 E	System error.
CPF3A12 E	Interface error.
CPF3A13 E	Access ID not valid.
CPF3C90 E	Literal value cannot be changed.
CPF3CF1 E	Error code parameter not valid.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API Introduced: V3R6

Top | Office APIs | APIs by category

Write to SNADS File Server Object (QZDWTFSO) API

Required Parameter Group:

1	File server object handle	Input	Char(32)
2	Operation requested	Input	Binary(4)
3	File server object data length	Output	Binary(4)
4	File server name structure	Input	Char(68)
5	Resume position	Input	Binary(4)
6	Buffer area	Input	Char(*)
7	Bytes provided in buffer area	Input	Binary(4)
8	Error code	I/O	Char(*)

Threadsafe: No

The Write to SNADS File Server Object (QZDWTFSO) API has four operations that can be used to write to an FSO. The first two operations, **write** and **terminate write**, are required to completely write to an FSO. The write operation copies the data from the caller's buffer into the file server object. This operation may be called multiple times to write all the data. The terminate write operation must be done after all the writes are completed. This operation closes the file server object and makes the FSO available to other processes to read it.

The two optional operations are **suspend write** and **resume write**, both of which are supported only by the SNADS general file server. These operations can be used to suspend the writing of a SNADS general FSO and then return to writing (resume) at a later time. These operations could be used in conjunction with the initiate read location operation.

An example of the usage of these operations follows. When data is being sent from one system to another, the sender on the source side is reading the file server object and the receiver is writing the data received into a new FSO on the target side. If the communications connection goes down during the middle of this data transmission, the receiver issues the suspend write operation. When the communications connection comes back up, the sender starts reading again using the initiate read location operation (part of the Read SNADS File Server Object (QZDRDFSO) API), and the receiver starts writing again using the resume write operation.

There are two types of file server objects that the QZDWTFSO API operates on: SNADS general file server objects and DIA file server objects. The interface to both of these types of objects is essentially the same. There are four SNADS general file server write operations and two DIA file server write operations provided by this API.

SNADS General File Server

The SNADS general file server provides a means for SNADS support to save the data object portion of a distribution request. The general file server stores objects in a byte stream format. No transformations are done to the data when it is written into or read out of a general file server object. General file server objects are owned by QSNADS and are stored in the QUSRSYS library.

The SNADS general file server can support objects up to 4 gigabytes. Each file server object space can only be

16MB large, but the SNADS general file server has the capability of chaining file server objects together. If the amount of data is greater than 16MB, the SNADS general file server creates as many file server object spaces as necessary and then chains them together. This function is transparent to the caller of the SNADS general file server APIs. After a chain of FSOs is created, this chain is still referred to by **one** file server object handle.

The following paragraphs describe each of the SNADS general file server write operations.

Write

The write operation copies the contents of the caller's buffer to the file server object. If the data being written is greater than 16MB, the write operation automatically creates new file server object spaces and chains them together. The caller may call the write operation multiple times to write all the data.

Terminate Write

Once this operation is called, no other write operations can be done on the specific file server object. Also, the file server object cannot be read or have an access ID assigned to it until the terminate write operation completes.

Suspend Write

The suspend write operation allows the caller to temporarily suspend the writing of an object. This operation does not assign an access ID to the file server object, nor does it increment the objects usage count.

Resume Write

The resume write operation does the setup needed to resume writing into a file server object. It is similar to the create file server object API since it needs to be called only once before write operations can be done. The caller of this operation must pass in the starting position of where to continue writing in the file server object. This number is based on how many bytes were written when the suspend write operation was called.

Note: The resume write operation also checks the system storage threshold to make sure there is enough room for the rest of the data to be written.

DIA File Server

The DIA file server provides a means for SNADS to save the data object portion of an OfficeVision distribution request. The DIA file server objects are of type *DTO (distribution tracking object), are owned by the QDOC user profile, and are stored in the QUSRSYS library.

The DIA file server can support objects up to 2 gigabytes. Each file server object space can only be 16MB large, but the DIA file server has the capability of chaining file server objects together. If the amount of data is greater than 16MB, the DIA file server creates as many file server object spaces as necessary and then chains them together. This function is transparent to the caller of the DIA file server routines.

The following sections describe each of the DIA file server write operations.

Write

The write operation copies the contents of the caller's buffer to the DIA document object. If the data being written is greater than 16MB, the write operation automatically creates new spaces and chains them together. The caller may call the write operation multiple times to write all the data.

Terminate Write

When the caller completes writing the entire document, the terminate write operation must be called to save the distribution tracking object. The DIA document will also be closed. The DTO and the document are not available to other processes until this operation is called. Once this operation is called, no other write operations can be done on the specific file server object. Also, the file server object cannot be read or have an access ID assigned to it until the terminate write operation completes.

Authorities and Locks

API Public Authority

*EXCLUDE

Required Parameter Group

File server object handle

INPUT; CHAR(32)

The necessary linkage to the file server object. The value of this parameter should be taken from the FSO handle that was returned on the call to the Create SNADS File Server Object (QZDCRFSO) API.

Operation requested

INPUT; BINARY(4)

The operation requested is used by the QZDWTFSO API to determine which operation, write, terminate write, suspend write, or resume write, the caller is requesting. Valid values for this parameter are as follows:

Write	5
Terminate write	6
Suspend write	7
Resume write	8

File server object data length

OUTPUT; BINARY(4)

The total length of the file server object that has been written. This parameter is returned on the terminate write, suspend write, and resume write functions.

File server name structure

INPUT; CHAR(68)

The file server name (either DIA or SNADS) and the server name length.

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	BINARY(4)	Length of file server name
4	4	CHAR(64)	File server name

The length of the file server name is the number of bytes in the file server name defined by the SNADS architecture. Valid values for length of the file server name are as follows:

4	SNADS general file-server name
4	DIA file-server name

The file server name is the name value defined by the SNADS architecture for the file server that is to be used. The only file servers available with this API are the SNADS general file server and the DIA file server. Valid values for file server name:

'21F0F0F6'X	SNADS general file-server name
'20F0F0F1'X	DIA file-server name

Resume position

INPUT; BINARY(4)

The place where the resume write function is to resume writing in the FSO. This parameter is ignored on the write, terminate write, and suspend write functions. Valid values for this parameter are as follows:

1 through the current file server object data length

Buffer area

INPUT; CHAR(*)

The data that the caller wants to write into a file server object.

Bytes provided in buffer area

INPUT; BINARY(4)

The number of bytes of data in the buffer area that are to be written to the file server object. Valid values for this parameter are as follows:

1 byte through 16 megabytes

Error code

I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3A0D E	Temporary server error.
CPF3A09 E	System error.
CPF3A1A E	Location in file server object not valid.
CPF3A1B E	File server object operation not valid.

CPF3A1C E	File server object request not valid.
CPF3A1D E	Limit of number of open file server objects exceeded.
CPF3A15 E	File server object not found.
CPF3A17 E	Permanent server error.
CPF3C90 E	Literal value cannot be changed.
CPF3CF1 E	Error code parameter not valid.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API Introduced: V3R6

Top | Office APIs | APIs by category

Word Separator Tables

The single-byte character set (SBCS) EBCDIC code page is extracted from the CCSID of the current job. The characters in the word list are mapped from the job's code page to the multinational code page 500 except for Greek and Turkish. Greek is mapped to code page 875; Turkish is mapped to code page 1026.

Delimiter Categories

Each character in a code page is assigned to a delimiter category (always, sometimes, and never a delimiter) as shown in the following tables.

Table 1. Always Delimiters

Category	Context-Dependent Definition	Description	Examples (CP500)
A	--	Basic delimiter[1]	Blank
D	--	Other delimiters	{) &
[1]This category can only contain 1 character.			

Table 2. Never Delimiters

Category	Context-Dependent Definition	Description	Examples (CP500)
L	--	Lowercase alphabetic characters	a b c
U	--	Uppercase alphabetic characters	A B C
N	-	Numeric characters, not including superscripts or subscripts	0 1 2

Table 3. Sometimes Delimiters

Category	Context-Dependent Definition	Description	Examples (CP500)
E	{ e | a < e < a }[1], [2]	Set of punctuation characters not treated as delimiters when surrounded by alphabetic characters	'
F	{ f | n < f < n }[3]	Set of punctuation characters not treated as delimiters when surrounded by numerics	. , : / -
G	{ g | p < g < n }[4]	Set of punctuation characters not treated as delimiters preceded by punctuation and followed by a numeric	. , $
H	{ h | ((p < h) and (h = p) }	Set of punctuation characters not treated as delimiters when immediately preceded by an identical character	. , : / - * = # %
I	{ i | F and H }	i is an element of F and H	. , : / -

J	{ j | E and F and H	j is an element of E and F and H	None
K	{ k | F and G and H	k is an element of F and G and H	. ,

Note:

1. a = any character in L or U
2. < means precedes
3. h = any character in N
4. p = punctuation (nonalphabetic and nonnumeric)

Considerations for the Sometimes Delimiters Table

Categories E through K are usually called possible delimiters because they function as delimiters only in certain contexts.

Characters . (period), ! (exclamation point), and ? (question mark) have a special status. When identical characters from this category occur together in a sequence, the individual characters do not act as delimiters; the entire sequence of characters forms a single token. However, the sequence of characters taken together does act as a delimiter because the sequence forms a token separate from characters that precede and follow it. For example, the text streams:

- "abc...def" is broken into three tokens: "abc", "...", and "def".
- "Unbelievable!!!!" yields two tokens: "Unbelievable" and "!!!!".
- "Astonishing!!!???" yields three tokens: "Astonishing", "!!!", and "???".

A simple token table is a 256-element array of unsigned characters. The simple token category value for the character is found in the element indexed by the code point of each character. Each code point (character) must be assigned exactly one category. If, according to the above definition of sets, a character is a member of more than one category, it should be assigned to the highest level category (for example, the category with the letter name latest in alphabetical order).

The categories assigned to each character in the three code pages are shown in the following tables. See the appendix that shows the code pages in globalization to refer to the characters that match these tables.

- Table 4 shows the simple token table for code page 500.
- Table 5 shows the simple token table for code page 875 (Greek).
- Table 6 shows the simple token table for code page 1026 (Turkish).

Table 4. Simple Token Table for Code Page 500

Hex Digits 1st > 2nd V	**4-**	**5-**	**6-**	**7-**	**8-**	**9-**	**A-**	**B-**	**C-**	**D-**	**E-**	**F-**

-0	A	D	I	L	U	D	D	D	D	D	D	N
-1	D	L	I	U	L	L	D	G	U	U	D	N
-2	L	L	U	U	L	L	L	G	U	U	U	N
-3	L	L	U	U	L	L	L	E	U	U	U	N
-4	L	L	U	U	L	L	L	D	U	U	U	N
-5	L	L	U	U	L	L	L	D	U	U	U	N
-6	L	L	U	U	L	L	L	D	U	U	U	N
-7	L	L	U	U	L	L	L	D	U	U	U	N
-8	L	L	U	U	L	L	L	D	U	U	U	N
-9	L	L	U	D	L	L	L	D	U	U	U	N
-A	D	D	D	I	D	D	D	D	D	D	D	D
-B	K	G	K	H	D	D	D	D	L	L	U	U
-C	D	H	H	D	L	L	U	D	L	L	U	U
-D	D	D	D	E	L	D	U	D	L	L	U	U
-E	D	D	D	H	L	U	U	E	L	L	U	U
-E	H	D	H	D	D	D	D	D	L	L	U	D

Table 5. Simple Token Table for Code Page 875 Greek Support

`Hex Digits 1st > 2nd V`	**4-**	**5-**	**6-**	**7-**	**8-**	**9-**	**A-**	**B-**	**C-**	**D-**	**E-**	**F-**
-0	A	D	I	D	D	D	E	G	D	D	D	N
-1	U	U	I	U	L	L	D	L	U	U		N
-2	U	U	U	U	L	L	L	L	U	U	U	N
-3	U	U	U	U	L	L	L	L	U	U	U	N
-4	U	U	U	D	L	L	L	L	U	U	U	N
-5	U	U	U	U	L	L	L	L	U	U	U	N
-6	U	U	U	U	L	L	L	L	U	U	U	N
-7	U	U	U	U	L	L	L	L	U	U	U	N
-8	U	U	U	U	L	L	L	L	U	U	U	N
-9	U	U	U	D	L	L	L	L	U	U	U	N
-A	D	D	D	I	L	L	L	L	D	D	D	D
-B	K	G	K	H	L	L	L	L	L	D	D	D
-C	D	D	H	D	L	L	L	L	L			
-D	D	D	D	E	L	L	L	L	L	E		
-E	D	D	D	H	L	L	L	L	D	D	D	D
-F	H	D	H	D	L	L	L	L	D	D	D	D

Table 6. Simple Token Table for Code Page 1026 Turkish Support

Hex Digits 1st > 2nd V	4-	5-	6-	7-	8-	9-	A-	B-	C-	D-	E-	F-
-0	A	D	I	L	U	D	L	D	L	D	L	N
-1	D	L	I	U	L	L	L	G	U	U	D	N
-2	L	L	U	U	L	L	L	G	U	U	U	N
-3	L	L	U	U	L	L	L	E	U	U	U	N
-4	L	L	U	U	L	L	L	D	U	U	U	N
-5	L	L	U	U	L	L	L	D	U	U	U	N
-6	L	L	U	U	L	L	L	D	U	U	U	N
-7	L	L	U	U	L	L	L	D	U	U	U	N
-8	D	L	D	U	L	L	L	D	U	U	U	N
-9	L	L	U	D	L	L	L	D	U	U	U	N
-A	U	U	L	I	D	D	D	D	D	D	D	D
-B	K	U	K	U	D	D	D	D	L	L	U	U
-C	D	H	H	U	D	L	D	D	D	D	H	D
-D	D	D	D	E	G	D	G	D	L	L	U	U
-E	D	D	D	H	D	U	D	E	L	L	U	U
-E	H	H	H	U	D	D	D	D	L	L	U	D

Notes:

(1)PCFILE is a file assigned the document type of PCFILE (Document Interchange Architecture type of 14) by the Client Access program.

(2)PCFILE is a file assigned the document type of PCFILE (Document Interchange Architecture type of 14) by the Client Access program.
