



O P E N P O R T™  
technology

**IPLP PROTOCOL COMMANDS  
FOR MESSAGING**

**DRAFT Version 1.0**

**990505**

© Copyright 1995, 1996, 1997, 1998 Open Port Technology, Inc.

All rights reserved.

This document contains confidential and proprietary materials and information of Open Port Technology, Inc. and its licensors, which are protected by copyright, patent and trade secret laws. No copy, disclosure, electronic reproduction or use of any portion of this document, or its content, may be made without the express written consent of Open Port Technology, Inc.

Open Port Technology, Open Port Technology logos, Open Port Harmony® NSP and Open Port Harmony® Pro are trademarks of Open Port Technology, Inc. All other marks used in this document are trademarks of their respective organizations. Some components of the Open Port Technology Harmony® product lines are covered by U.S. Patent Nos. 5,369,686 and 5,712,907. Other patents are pending.

**CHAPTER 1 IPLP PROTOCOL GENERAL INFORMATION**

About This Chapter	4
Protocol Structure	5
IPLP Protocol Native Data Types	7
IPLP Protocol Acronyms	8
Common Status Codes	9
IP LaunchPad Protocol Attributes	10

**CHAPTER 2 IPLP PROTOCOL COMMANDS FOR MESSAGING**

About This Chapter	16
HELLO	17
FILE	19
FILE SUBCMD="FABORT"	20
FILE SUBCMD="FCLOSE"	21
FILE SUBCMD="FCOMMIT"	22
FILE SUBCMD="FCREATE"	23
FILE SUBCMD="FGET_ATTRIBUTES"	25
FILE SUBCMD="FOPEN"	27
FILE SUBCMD="FREAD"	28
FILE SUBCMD="FSET_ATTRIBUTES"	30
FILE SUBCMD="FWRITE"	32
FILE SUBCMD="MAKE_PERSISTENT"	33
FOLDER	34
FOLDER SUBCMD="ENUM_CONTENTS"	35
FOLDER SUBCMD="ENUM_FOLDERS"	37
MESSAGE	39
MESSAGE SUBCMD="ABORT"	40
MESSAGE SUBCMD="ADD_ATTACHMENT"	41

MESSAGE SUBCMD="ADD_DESTINATION"	42
MESSAGE SUBCMD="COMMIT"	43
MESSAGE SUBCMD="CREATE"	44
MESSAGE SUBCMD="DELETE"	46
MESSAGE SUBCMD="FORMAT"	47
MESSAGE SUBCMD="GET_ATTACHMENTS"	49
MESSAGE SUBCMD="GET_ATTRIBUTES"	51
MESSAGE SUBCMD="GET_STATUS"	53
MESSAGE SUBCMD="SET_ATTRIBUTES"	55
MESSAGE SUBCMD="SUBMIT"	57

CHAPTER 3      **SAMPLE MESSAGING SESSION**

Sample Messaging Session	60
--------------------------	----

CHAPTER 2

# **IPLP PROTOCOL COMMANDS FOR MESSAGING**

This chapter documents a subset of the IP LaunchPad protocol commands. This command set contains the commands needed to:

- Initiate a session with the IP LaunchPad Client Access Module connector.
- Create and process files
- List folder names and folder contents
- Create, process and submit messages

## About This Chapter

This chapter contains the following sections:

**HELLO** Documentation of the HELLO command.

**FILE** Documentation of the FILE command and its subcommands.

**FOLDER** Documentation of the FOLDER command and its subcommands.

**MESSAGE** Documentation of the MESSAGE command and its subcommands.

### Command Definitions

Each command is defined in the following way:

**Description** Brief overview of the command's function.

**Command** Example command, with variables in italics and optional parameters in square brackets.

**Parameters** Definitions of all command parameters, including optional parameters and default value (if any). If there are no command parameters, this section is omitted.

**Special STATUS Response** The possible response to this particular command from the receiving node. See XX for a list of general responses that could be returned for most IP LaunchPad Protocol commands.

**Example** An example command and status response pair with sample parameter values.

**Other** As required, additional sections provide more detail on command parameters.

# MESSAGE

## Description

The MESSAGE command set is used to create messages for sending.

## Subcommands

The FILE command set includes the following subcommands:

- ABORT (page 40)
- ADD\_ATTACHMENT (page 41)
- ADD\_DESTINATION (page 42)
- COMMIT (page 43)
- CREATE (page 44)
- DELETE (page 46)
- FORMAT (page 47)
- GET\_ATTACHMENTS (page 49)
- GET\_ATTRIBUTES (page 51)
- GET\_STATUS (page 53)
- SET\_ATTRIBUTES (page 55)
- SUBMIT (page 57)

## MESSAGE SUBCMD="ADD\_ATTACHMENT"

### Description

The command attaches a pre-existing file to the current message.

### Command

```
MESSAGE SUBCMD="ADD_ATTACHMENT" FID=&file_id  
[ SMID=#submessage_id]
```

### Parameters

**FID** Datatype: GUID

The GUID of the file that is to be attached to the specified message.

**SMID *optional*** Datatype: GUID

The Sub-Message ID of a single destination of the specified message. If one or more SMID= parameters are included with the ADD\_ATTACHMENT subcommand, the specified file will be attached ONLY to the messages sent out to the specified destination(s). If there are no SMID= parameters included, the file will be attached to all instances of the outbound message.

### Special STATUS Response

There are no special return codes for this subcommand. (Refer to page 9 for a list of response codes for all MESSAGE subcommands.)

### Example

```
MESSAGE SUBCMD="ADD_ATTACHMENT FID=&{CCCCCCCC-  
CCCC-CCCC-CCCC-CCCCCCCCCCCC} SMID=&{DDDDDDDD-  
DDDD-DDDD-DDDD-DDDDDDDDDDDD}  
B: STATUS CODE=#200
```



## MESSAGE SUBCMD="CREATE"

### Description

Creates a new message on the server. All other operations (setting attributes, adding destinations, adding attachments, etc.) are performed with other MESSAGE subcommands.

### Command

```
MESSAGE SUBCMD="CREATE" [ FOLDER_ID=&folder_id]
```

### Parameters

**FOLDER** *optional* Datatype: GUID

The GUID of the folder in which the message job will be created. If this parameter is omitted, the job will be created in the current user's "Outbox" folder.

---

*When a new message is created on the messaging server, several attributes are automatically created for it, using data from the sender's personal profile stored on the User Server. (Examples to follow)*

---

---

## Special STATUS Response

The server returns one of the following STATUS CODE= values in response to a MESSAGE SUBCMD="CREATE" command:

CODE	MEANING	PARAMETERS
200	The command executed successfully.	MID= <i>message_id</i> (datatype: GUID) The GUID of the newly created message. This is also the name of the transaction and will be used in all subsequent commands associated with this message.  The client may issue multiple CREATE CLASS=MESSAGE commands in order to have multiple transaction open simultaneously. Each transaction is handled separately by the MBS.

### Example

The following example creates a message from the current user in a particular folder:

```
A: MESSAGE SUBCMD="CREATE" FOLDER_ID=&{AAAAAAAAA-  
AAAA-AAAA-AAAA-AAAAAAAAAAAA}  
B: STATUS CODE=#200 MID=&{BBBBBBBBB-BBBB-BBBB-  
BBBB-BBBBBBBBBBBBBB}
```

## MESSAGE SUBCMD="GET\_ATTACHMENTS"

### Description

Uploads a list of an inbound message's attachments.

### Command

```
MESSAGE SUBCMD="GET_ATTACHMENTS" MID=&message_id  
  [SMID=#submessage_id] [FILE_TYPE="file_type"]  
  [FILE_ENCODING="file_encoding" ]
```

### Parameters

**MID** Datatype: GUID

The ID of the message about which you are seeking information.

**SMID** Datatype: Integer

(Optional) The number that indicates one of the destinations associated with the specified message. To retrieve the list of attachments for an inbound message, you would use the value "1". If this is the case, then this parameter can be omitted. (You can also specify the MID of an outbound message and the SMID of one of that message's destinations, to check on what attachments were being sent to that destination.)

**FILE\_TYPE** Datatype: String

(Optional) A string representing the mime type of the file. (Example: "image/tiff"). If this parameter is included, the command will only upload those attachments whose filetypes match the specified filetype. If this parameter is omitted, the command will upload the files in their original formats. (Examples: if the message is being uploaded to an Email mailbox, the attachments can be delivered in their original formats. If the message is being delivered to a fax machine, on the other hand, this command can grab ONLY the TIFF versions of the attachments.)

**FILE\_ENCODING** *optional* Datatype: String

(Optional) A string representing the mime encoding of the file. (Example: "8bit") If this parameter is omitted, the command will upload the files in their original formats.

### Special STATUS Response

CODE	MEANING	PARAMETERS
200	The command executed successfully.	FIDS= {&{GUID}, &{GUID}, ...} (datatype: set) A set consisting of the GUID(s) of the uploaded files. Even if there is only one FID, it will be returned enclosed in a set of curly brackets ("{"}")

### Example

```
A: MESSAGE SUBCMD="GET_ATTACHMENTS "  
MID=&{AAAAAAAA-AAAA-AAAA-AAAA-AAAAAAAAAAAA}  
SMID=#1 FILE_TYPE="image/tiff"  
B: STATUS CODE=#200 FIDS={&{BBBBBBBB-BBBB-BBBB-  
BBBB-BBBBBBBBBBBB}, &{CCCCCCC-CCCC-CCCC-CCCC-  
CCCCCCCCCCCC}}
```

## MESSAGE SUBCMD="GET\_ATTRIBUTES"

### Description

Retrieves the values of one or more attributes to a message.

### Command

```
MESSAGE SUBCMD="GET_ATTRIBUTES" MID=&message_id  
SMID=&submessage_id  
ATTRIBUTE_NAMES={"attribute_1","attribute_2"...}
```

### Parameters

**MID** Datatype: GUID

The ID of the message whose attribute values are to be retrieved.

**SMID** Datatype: Integer

The number that indicates one of the destinations associated with the specified message. If one or more SMID= parameters are included with the GET\_ATTRIBUTES subcommand, the command will retrieve ONLY those values for the specified destination. If there are no SMID= parameters included, the command returns the general set of attributes that apply to all destinations.

**ATTRIBUTES** Datatype: Set

A list of the attributes for which the function should retrieve values. Each attribute name should be enclosed in double quotes, and separated from the next name by a comma. The entire list must be enclosed in curly brackets ("{}").

See page 11 for a list of message attributes.

---

*When a new message is created on the messaging server, several attributes are automatically created for it, using data from the sender's personal profile stored on the User Server. (Examples to follow)*

---

---

### Special STATUS Response

CODE	MEANING	PARAMETERS
200	The file was created successfully.	ATTRIBUTE_VALUES= {value_list} (Datatype: Set) The list of values for the specified attributes, in the same order as the attribute list given in the function call. Each value is surrounded by double quotes, and separated from the next value by a comma. The entire list is enclosed in a set of curly brackets ("{"")

### Example

```
A: MESSAGE SUBCMD="GET_ATTRIBUTES"  
MID=&{AAAAAAAA-AAAA-AAAA-AAAA-AAAAAAAAAAAA}  
SMID=#2 ATTRIBUTE_NAMES={"PAGES", "FILE_SIZE"}  
B: STATUS CODE=#200  
ATTRIBUTE_VALUES={"2", "5673"}
```