Open Messaging System

Batch User Agent Reference Guide

Version: 5.0
Date: June 2013

Contents

Contents

Tables:

Preface

About this manual

This manual is a guide to the OpenMS Batch User Agent. OpenMS is an electronic messaging system, which complies with the CCITT X.400 and X.500 standards for Message Handling Systems and Directory Services. The Batch User Agent is the mechanism, which provides host access to the X.400 Message Store and the X.500 Directory.

Purpose

The primary purpose of the manual is to describe the commands used in applications, which communicate with OpenMS. It also provides a general overview of OpenMS and of message handling systems in general.

Target group

The manual is chiefly intended for application developers who are familiar with message handling system technology.

Summary of contents

The main part of the manual is divided into five chapters. These are followed by 7 appendices and a glossary. Experienced users may wish to skip chapters 1 and 2.

- Chapter 1, *Overview*, provides a general introduction to X.400-based Message Handling Systems and X.500-based Directory Services and an overview of the Batch User Agent.

- Chapter 2, *Messaging concepts*, examines various concepts that are fundamental to messaging systems, such as message format, message addressing and attributes, and message processing and validation.

- Chapter 3, *Structures and settings in the Batch User Agent*, discusses the parameters that govern how the Batch User Agent operates: user profiles, time formats and character sets and the use of folders.

- Chapter 4, *Getting started*, briefly explains how OpenMS works, how to write and transfer command script files for the Batch User Agent, and gives examples of sending and reading messages.

- Chapter 5, *Command descriptions*, is in two parts. The first part deals with various general aspects of the command set, such as syntax and notational conventions, while the second comprises detailed descriptions of all the available commands in alphabetical order.

- The appendices consist of lists of system limits, error messages, diagnostic codes, message types, file formats and various strings (numeric strings, printable strings and so forth). They also provide a brief overview of the system administrator tasks relevant to the Batch User Agent.

- The glossary provides brief descriptions of important terms and includes a list of the chief abbreviations used in the manual and in related publications.

1   Overview

OpenMS is an X.400-based electronic messaging system that features X.500-compliant Directory Services. This chapter, which experienced users can feel free to skip, provides overviews of the CCITT X.400 and X.500 recommendations and a general introduction to the OpenMS Batch User Agent.

1.1   X.400-based Message Handling Systems

The recommendations and standards in the CCITT X.400 series define a set of protocols describing a Message Handling System which enables users to exchange data on a store-and-forward basis. The store-and-forward principle makes data exchange independent of the equipment which the communicating parties use and of the time at which they send or retrieve the information.

The Message Handling System defines an *interpersonal message* (IPM) service. The original CCITT Message Handling System recommendations date from 1984, the revised definition from 1988; thus the types of interpersonal message supported by the OpenMS software are referred to as *IPM84* and *IPM88*.

The Message Handling System (MHS) model can usefully be viewed as having three elements: the *User Agent* (UA), the *Message Transfer System* and the *Message Store*. User agents form the interface between the actual users and the rest of the Message Handling System. The Message Transfer System comprises one or more Message Transfer Agents (MTAs) which are responsible for message routing and delivery. The Message Store is an optional component that acts as an intermediary between a User Agent and its local Message Transfer Agent. The MHS model also includes an access unit mechanism, which provides for intercommunication between the Message Handling System and other communication systems or services.

In outline, the Message Handling System works in the following way: One user, known as the originator, submits a message to the Batch User Agent. The Batch User Agent passes the message to the Message Transfer System. A Message Transfer Agent then delivers the message either to the intended recipient's User Agent or to a Message Store from which it can be retrieved by the recipient's User Agent. The recipient can then read and process the message.

OpenMS fits into this model by providing a Batch User Agent and a Message Store along with additional functionality, which extends the basic message handling capabilities.

1.2   X.500-based Directory Services

The CCITT X.500 Directory Service recommendations are designed to allow users to conduct name and address searches on a global basis. A user wishing to look up name and address details for a potential recipient requests a Directory User Agent to perform a Directory lookup. This is intended to make message addressing user-friendlier by allowing entire complex X.400 addresses to be retrieved by way of a simple common name.

In OpenMS, the User Agent servers are capable of acting as X.500 Directory User Agents. That means that users can access X.500 services from a terminal or a terminal emulation without the need for a dedicated Directory User Agent.

1.3   About OpenMS and the Batch User Agent

The OpenMS messaging system combines X.400-compliant Message Store and User Agent functionality and X.500 Directory User Agent functionality with a range of extended capabilities.

1.3.1   Components of OpenMS

OpenMS has a modular architecture. It has six principal modules:

- Batch User Agent

- Message Store

- EDI Server

- P7 Server

- CDIF Server

- Local User Agent

For the purposes of this manual, the three most important components are the Batch User Agent, the Message Store and the EDI Server.

The Batch User Agent

The Batch User Agent is a batch mode interface to the Message Store. In batch mode, instead of entering commands at the prompt the user stores them in a file and transfers this file to a directory on the OpenMS system. This directory is scanned periodically, and when at least one command file is found it is passed to the Batch User Agent for processing. Chapter 5 of this manual describes the Batch User Agent command set.

The Message Store

The Message Store is a database where messages and documents are stored and exchanged. It maintains user-specific mailboxes in which messages are stored. It interacts with the Message Transfer System (see section 1.1 above), either forwarding outgoing mail or retrieving incoming mail.

The OpenMS Message Store is designed to handle a large number of subscribers, many simultaneous users, a large number of stored messages, high message throughput and a practically unrestricted message size.

The EDI server

The EDI server supports the uploading and downloading of EDI documents and Transmission Sets. EDI (Electronic Data Interchange) is a set of messaging standards, which define how formatted documents can be exchanged on an application-to-application basis, rather than on an interpersonal basis as is the case with electronic mail. The main components of the EDI server are the Trading Relations database and the Transmission Set Processor. For more information on EDI processing see section 2.5, *EDI processing and validation*, on page 12).

Other components of OpenMS

The P7 server supports the operations defined in the CCITT X.413 Message Store Access Protocol (generally known as the P7 protocol). It is designed to support User Agent applications using P7-compliant command syntax.

The CDIF server provides access to the Message Store for clients, primarily personal computers, supporting the CDIF (Common Document Interchange Format) protocol. The commands of the CDIF command set are not described in this manual.

The Local User Agent is the user's interface to OpenMS. It accepts the user's commands, performs correctness and security checks, and then interacts with the Message Store.

Interaction between the Batch User Agent and the Message Store

The way in which the Batch User Agent and the Message Store interact is perhaps best described by way of an analogy.

When you want to send someone a letter by post, what do you do? You write the letter, perhaps you add some newspaper cuttings and photographs, you put everything together in an envelope, write the name and address on the envelope, then put the whole thing in a mailbox or take it down to the post office and hand it over at the counter. The letter stays there until the next collection and is then sent on its way by road, rail, air or bicycle, or on foot.

OpenMS works in much the same way. Using the commands of the Batch User Agent interface, you compose your message from text you type in an editor and from existing files and messages, and you specify the name and electronic address of the message recipient. The Batch User Agent then passes the message(s) to a Message Transfer Agent in the Message Transfer System, which delivers it as soon as possible to the recipient's Message Store.

Now consider a slightly different analogy. You have a P.O. box at a post office, and every so often you go along to pick up all the mail that has accumulated there for you. Similarly, the Message Store stores incoming electronic mail, and you can periodically look to see whether any mail has arrived for you.

1.3.2   Accessing the Batch User Agent

You access the Batch User Agent by placing files in specific directories on the OpenMS computer. The system scans for files on that directory periodically and the Batch User Agent is started, if there are files to be processed. You transfer the files using standard file transfer protocols. These are discussed in detail in section 4.4, "Host access"

1.3.3   System requirements

An appropriate file transfer utility (BSC, FTAM, FTP, etc.) must be installed on your system. This utility must be correctly configured on the host system and on your own system. See section 4.4 on page 21 for details of the use of the various file transfer protocols.

1.3.4   Data security

Each user has private mailboxes on the OpenMS server. Access is protected by a unique user name and a password, so that no other user has access to your data.

A system user name and password are required by the file transfer protocol used. In order to minimize administration overhead, it is recommended that the OpenMS account name and password should be the same as the system account name and password.

2 Messaging concepts

This chapter, which experienced users can feel free to skip, discusses some of the fundamental aspects of electronic messaging systems, such as message format, message addressing, message attributes, reports and notifications.

2.1 Basic concepts

The essential purpose of an electronic messaging system is to allow information to be exchanged between users. This entails one user sending a *message* and another user retrieving it. The message handling system sends back *reports* on the progress and delivery of the transferred message; and the User Agent may send back *notifications* indicating whether the message has been read. Let us examine these terms in more detail.

2.1.1 What is a message?

Put simply, a message is the electronic equivalent of a "paper" letter that you send by the postal mail system. As defined in the CCITT X.400 recommendations, a message essentially consists of an *envelope* and a *content*.

The *envelope* supplies information that the Message Transfer System needs in order to be able to deliver the content to the right place. This information identifies the message originator and the intended recipients, documents its passage through the Message Transfer System, and characterizes its content.

The *content* is the information which one user wishes the other to receive. The messaging system simply transfers this without modifying or examining it. The content is further divided into two types of information: a *heading* and a *body*. The heading comprises memo-like information such as the message importance, the subject and the recipient list, while the body consists of one or more freely combinable body parts, such as directly entered text or binary files.

2.1.2 What are reports and notifications?

A report is information returned by the Message Transfer System to let the originator know whether a message has been delivered to the specified recipient's address as intended. Depending on what instructions the Batch User Agent has been given, this information may be returned (by the Message Transfer Agent) when the message is delivered or only if the message fails to be delivered correctly. These two types of report are known as delivery and non-delivery reports respectively.

The originating user may also ask to be notified whether or not a message has been retrieved and read by the intended recipient. These notifications (sent by the recipient's User Agent) are known as receipt and Non-Receipt notifications.

2.2 Message composition

When you prepare a message for transmission (using the Batch User Agent's *SUBMIT* command), there are essentially two activities involved. First you define the heading information, then you specify the components that are to form the body of the message (the body parts).

The heading information defines the message priority (e.g. *urgent*), the message subject (freely definable) and whether the message is to be sent immediately or deferred to some later time. Most importantly, it defines the intended recipients; and this information is also used in the transfer envelope. Message addressing is dealt with in the next section.

The body consists of one or more body parts. The Batch User Agent supports two types of body part: input from the command script and uploaded text and binary data. Forwarding of messages is only supported for incoming messages.

Input from the command line refers to text which you enter directly as part of the *SUBMIT* command in the command script following the keyword *TEXT*. You terminate text input with the keyword *EOTEXT*.

Uploaded data (also referred to as attached data) is data which is uploaded from the user's system to form a body part in the outgoing message. The data may be in the form of text in the IA5 character set (International Alphabet Number 5) or the ISO Latin 1 character set (see the description of the SUBMIT command on page 87ff.), or it may be a binary file. The name of the uploaded file or files is specified with the *TEXTFILE* or *FILE* keyword in the *SUBMIT* command.

2.3 Message addressing

The envelope which accompanies the message content carries a range of information that the Message Handling System requires in order to be able to transfer the message. Among the chief items of information in this transfer envelope are those relating to message addressing. With the OpenMS Batch User Agent, these specifications are taken from the *IMPDU_HEADING* section of the *SUBMIT* command (see page 87ff.).

An X.400 address comprises a number of components, which can be combined in different ways to identify a recipient uniquely.

Three different address forms are supported under the Batch User Agent. These correspond to the CCITT X.402 recommendations and are described in detail in the sections on the *SUBMIT* command (see page 87) and the *FETCH* command (see page 45).

The most important elements of an X.400 address are as follows:

COUNTRY	one of a predefined set of country codes identifying the country (e.g. DE for Germany)
ADMD	an administrative management domain (PTT authority) within the country
PRMD	a private management domain (non-PTT organization) within the country and subject to the ADMD
SURNAME	the surname of the intended recipient
GIVENNAME	the given name (first name) of the intended recipient
ORGANIZATION	the name of the organization at which the intended recipient can be reached

ORG_UNIT	the name of a subdivision of *ORGANIZATION*
COMMON_NAME	(IPM88 only) the defined full name of the intended recipient, typically a given name followed by a blank and a surname
UA_CONT_ID	the unique User Agent identification number of the intended recipient
X121_ADDRESS	the X.121 address of the intended recipient's terminal (a telephone number)
FREEFORMNAME	a free-form name which is not used for routing
DDA specification	a domain-defined attribute comprising a type and value, typically identifying a particular service available within the selected management domain.

2.4   Message attributes

In addition to the address data for the intended recipients, the transfer envelope carries information defining a range of other message attributes. These identify a message within the message handling system. They include indications of the message priority, the date and time of message creation, the User Agent entry type (see page 124) and entry status, the message subject, an interpersonal message identifier and a User Agent Content identifier.

Most of these attributes can be used to select messages in various commands. For details, see the section on the *selector* specification on page 78.

The entry status of a message is assigned by the system and can have one of four values: NEW, LISTED, OLD and SENT. These values have the following meanings:

– **NEW**

All received messages (including reports and notifications) are initially assigned the entry status NEW. This status applies until the message is listed or fetched by the user.

– **LISTED**

The entry status of received messages (including reports and notifications) changes from NEW to LISTED after the user has issued an appropriate LIST <message folder> command and the message has been included in the resulting list.

This status applies until the message is fetched or deleted by the user.

– **OLD**

The entry status of received messages (including reports and notifications) changes from NEW or LISTED to OLD after the user has issued an appropriate FETCH command. This status applies until the user deletes the message.

When the entry status of a message changes to OLD, the User Agent issues any receipt notification, which may have been required by the person sending the message.

– **SENT**

All messages successfully transmitted are assigned the entry status SENT. This status applies until the user deletes the message.

2.5   EDI processing and validation

An EDI document is a Transmission Set encoded in accordance with the ISO 9735 standard for Electronic Data Interchange for Administration, Commerce and Transport (EDIFACT). A

Transmission Set is a sequence of Interchanges. An Interchange is defined as a structured set of messages (user data) and service segments. The user data is the information content of the document, while the service segments consist of various headers and trailers, which encapsulate the user data and identify the sender, the recipient and other information required for message transfer. The Batch User Agent's EDI support mechanism enables EDI users to communicate with each other via X.400 without having to know anything about X.400 address and message syntax. Address translation and message packing and unpacking are handled automatically.

The Batch User Agent command set includes commands for uploading and downloading EDI documents. These commands are available only to users who have an EDI mailbox (EDI box). They access an EDI server which has two main components, the Transmission Set Processor and the Trading Relations database. For details of mailboxes, see section 3.4 on page 16.

To send an EDI document you first have to use the *UPLOAD* command (see page 100). This command does not convert or parse the EDI data in any way, but simply stores it in a temporary file, the Transmission Set, which is then processed in the background by the Transmission Set Processor (TSP). The TSP performs validation and address lookup by referring to the Trading Relations database, splits the transmission into its component parts (Interchanges) and then packs each Interchange into a separate X.400-style message. The messages are then submitted to the Message Transfer Agent and delivered to the recipient's Message Store. There, again after validation against the Trading Relations database, the X.400 header information is stripped and the Interchanges are placed in a Transmission Set. The *DOWNLOAD* command (see page 42) is then used to download the Transmission Set to the recipient's system.

Given two EDI partners - one acting as a sender and the other one as a receiver - the corresponding relation says whether there is a valid agreement between the two of them and, if so, what this agreement is like.

EDI Partner

An EDI partner is someone who is able to send and/or receive EDI documents. Each EDI partner is uniquely identified by the following attributes:

- a EDI partner identification
- a EDI partner identification qualifier

EDI Relation

An EDI relation is a directed relationship between a sending and a receiving EDI partner. Directed means, that the kind of relationship is not necessarily the same, if the sender and receiver change roles. (This is another trading relation, which is - at least theoretically - completely independent from the previous one.) However, an EDI relation determines whether the sending EDI partner is allowed to transmit EDI documents to the receiving one, or not.

EDI account (EDI ID)

Every EDI partner can have one or more difference EDI accounts (EDI ID)

Partnership Type

There are two kinds of EDI partnerships, which differ as the entry in his EDI account
Closed user group: YES or NO

Partner Type	Description
Open	An open EDI partnership is characterized as follows: EDI documents are generally accepted from any sending EDI partner.. Any EDI partner were able to send messages to this EDI Account
Closed	A closed EDI partnership is characterized as follows: No EDI documents are accepted at all, but certain senders may be excluded explicitly, i.e. allow them to send EDI documents to the closed EDI Account Only configured partner were able to send messages to this EDI Account

3   Structures and settings in the Batch User Agent

This chapter discusses the parameters that govern how the User Agent operates: user profiles,
time formats and character sets, and the use of folders.

3.1   User profiles

The OpenMS user administration subsystem includes a user database. The information stored in
this database includes a series of user profiles, one per user (or User Agent), which define
various user-specific parameter settings. These user profiles are set up by the system
administrator when he/she sets up the BUA login.

Among the settings defined in your user profile are:

- your password

- the type of mailbox you have been assigned

- the character set you use

The Batch User Agent only allows you to view and modify one of these settings, namely the
number of days for which EDI documents are archived. See the *LIST PROFILE* (page 70) and
REGISTER (page 75) commands for further details.

Refer to Appendix B on page 105 for details on all the settings in the profile which affect
operation of the Batch User Agent.

3.2   Time format

The Batch User Agent uses a UTC time format indicating the local user time (not GMT). This
is a 15-character string in the form "*YYYYMMDDhhmmss*Z".

The first 14 characters in the string are numeric and indicate the year, month, day, hours,
minutes and seconds. The final character is always a Z (uppercase or lowercase).

Only the values 1990 through 2089 are supported for the year specification.

3.3   Character sets

Internally, OpenMS uses the ISO Latin 1 character set. On your system you can use DECmcs (local VMS users), IA5IRV, IA5NRV, IBMPC (PC users) or ISO Latin 1. When transmitting data, OpenMS can handle IA5, ISO Latin 1 and T.61. Conversion is handled automatically.

The character set you use on your system is defined in your user profile. This character set is used for the command script, the result file, text attachments and forwarded attachments. .

The most common character set is IA5(IRV/NRV) and will be sufficient in 90 % of the cases. The variant NRV contains some country specific characters like German Umlaut.

For some special cases (German Umlaut, and other special signs) you may use the ISO Latin 1 character set (for IPM88 messages).

3.4   Mailboxes and folders

Messages that are sent to you are stored on arrival in a private mailbox. There are two types of mailbox, the standard mailbox (or MAIL box) for ordinary X.400 mail users and the EDI mailbox for EDI users (see section 2.5). Some Batch User Agent commands are available only to users with a particular type of mailbox. *UPLOAD*, for example, is available only to EDI users, while *SUBMIT* is available only to users with a standard mailbox.

Your mailbox in the Message Store comprises a number of message storage areas known as folders or infobases. There are a number of system folders and, for MAIL box users, sixteen user-definable (private) folders. The names of the system folders cannot be changed by the user. The names are:

- *INFOLDER* (MAIL box - for incoming mail messages, EDI box - for incoming EDI documents)

- *SENTFOLDER* (MAIL box - stores outgoing mail messages, EDI box - for outgoing EDI documents)

- *MAILBOX* (EDI box only - contains the various reports)

- *PROFILE* (EDI box only - contains the database information for the user profile)

- *SUBSCRIBER* (MAIL box only - contains address information for other subscribers)

- *FOLDER* (contains information on all the system and user folders)

The user-definable (private) folders, available to MAIL box users only, have default names (*FN01* through *FN16*). You can use the *MODIFY* command (see page 73) to assign your own names to these folders.

To move a message from one folder to another, MAIL box users can use the *FILE* command (see page 63). There are no restrictions on moving messages from one folder to another. Virtually any message can be moved to any folder. This allows you to keep all the messages related to a particular business partner within a single folder, regardless of message type or status.

To delete one or more messages from a folder you use the *DELETE* command (see page 40). You cannot delete messages with an entry status of *NEW*.

3.5   Logical directories

In the Batch User Agent, the various directories and subdirectories in which the different file types are stored are referenced by logical names. The following table shows a complete list of these logical names and indicates the file types stored in each of the directories.

BUA logical directory	File types	Description
BUA_LOGIN_DIR	*.bua	BUA command scripts. These contain the commands to be processed by the Batch User Agent.
BUA_LOGIN_DIR	*.ebc	EDI Transmission Set files for EBCDIC users. These files are stored here temporarily prior to transmission.
BUA_LOGIN_DIR	*.ets	EDI Transmission Set files for ASCII and other users. These files are stored here temporarily prior to transmission.
BUA_LOGIN_DIR	*.*	Temporary files. These could, for instance, be attachments, which are stored here temporarily prior to transmission.
BUA_RTS_DIR	*.ebc	EDI Transmission Set files containing the result of an EDI download (EBCDIC users only).
BUA_RTS_DIR	*.rts	EDI Transmission Set files containing the result of an EDI download (ASCII and other users).
BUA_RESULT_DIR	*.res	BUA result files. These contain the results of the corresponding *.bua files.
BUA_TXT_DIR	*.<ddd>	These files contain text body parts and are generated as a result of the FETCH command.
BUA_TXT_ATT_DIR	*.<ddd>	These files contain text attachments and are generated as a result of the FETCH command.
BUA_FOR_ATT_DIR	*.<ddd>	These files contain attached interpersonal messages and are generated as a result of the FETCH command.
BUA_BIN_ATT_DIR	*.<ddd>	These files contain binary body parts and are generated as a result of the FETCH command.

Table 1: BUA logical directories and file types

<ddd> is an alphanumeric string, which is incremented for each file generated with the *FETCH* command. The counter is initialized to "001" at login time and consists of exactly three alphanumeric characters incremented according to the following pattern ("001" - "009", "00a" - "00z", "010" - "019", "01a" - "zzz").

4   Getting started

In this chapter we give you a brief introduction to how the OpenMS interface works. We examine the structure of the command files, show you how to pass command files to the Batch User Agent using the various file transfer protocols available, and we work through some brief examples demonstrating how to send and read messages.

4.1   How the interface works

The Batch User Agent is a component, which provides host access to OpenMS. Host access allows subscribers to access their mailbox using a batch user interface on the OpenMS system. The batch user interface uses batch mode to send and receive messages to and from other subscribers.

In batch mode the subscriber does not communicate interactively with the host system by entering commands. He uses a file containing the necessary Batch User Agent (BUA) commands and separate pay load files of messages to be sent. Subscribers create the command file on their own systems using a text editor or an application and then copy the file to their own BUA login directory on the OpenMS system. The command script file must have the file extension ".bua". The general structure of command scripts is explained in section 4.2, "Command scripts" and the individual commands are described in detail in chapter 5.

Files are copied from the subscriber's system to the OpenMS system using a file transfer utility, which also provides access protection. The transfer protocols available and the way in which they are used are described in section 4.4, "Host access".

The subscriber's BUA login directory (see section 3.5 on page 17) on the OpenMS System is scanned periodically for command scripts having the file extension ".bua". (For information about frequency of scanning operations please contact your service support personnel.) When command script files are found, the Batch User Agent is started and the command script files are processed in order of their creation times. This means that the subscriber does not have to run the command scripts explicitly. They are processed through an external program, which calls the BUA if there are files to process.

The maximum number of command files to be processed in one go by the OpenMS BUA process is adjustable by service provider (For information about the number of command files please contact your service support personnel) and configured in parameter max_files_per_session of the configuration file (OMS_POLLER_DAT:OMS_POLLER_CNF.TXT, see Batch User Agent Host Access System Administration Manual).

If after a SUBMIT oder UPLOAD Command in a BUA script, the attachment is not present in the LOGIN-DIRECTORY, the BUA-process will recheck within five minutes whether an attachment has appeared. If after this interval an attachment is still not present, the BUA-process will create a error message and recheck for attachment after that in 10 second intervals. This algorithm will be reinitiated after a successful SUBMIT or UPLOAD.

The results of the processed command script files are written to a result file and stored in the subscriber's result file subdirectory (see section 3.5 on page 17). The subscriber can copy the result files back to his/her system and analyze the data. As soon as the ".bua" file has been processed and the result file has been written to the BUA result subdirectory, the command script file is deleted from the subscriber's BUA login subdirectory. The general structure of result files is explained in section 4.3, "Result files" and the result syntax for individual commands is described in detail in chapter 5.

There is no notification to inform the subscriber that a command script has been processed and that the result file has been generated. The subscriber has to wait an appropriate period of time

before retrieving the result file. The filename of the result file is the same as the filename of the command script but has the file extension ".res" instead of ".bua".

The pay load of fetched messages will be stored in appropriate subdirectories (see section 3.5 on page 16) and the reference for this file names will written into result file.

4.1.1   BUA temporary directory

Processing the command scripts, the BUA create for every subscriber a temporary sub directory (TMP) under the local OMS_POLLER_DIR The files (command script, result file and journal file) will be created under this temporary sub directory during the processing and deleted after the BUA processing is finish.

Note:
– In order to allow the Batch User Agent to process command scripts correctly, all additional files, such as attachments or EDI Interchanges, have to be transferred <u>before</u> the command script file, as the BUA cannot check for the presence of these files on the OpenMS system before processing the command script. Thus an error will be reported in the result file if a required file is not found.

– It is recommended that a copy of command scripts containing frequently used commands be kept on the host system so that they may be reused. The scripts can be copied to the subscriber's login directory on the OpenMS host system whenever they are required.

– It is also recommended that the result files be copied back to the originator system within 24 hours, since they will be stored in the BUA result subdirectory for a limited time only. For more information, please contact your service support personnel.

4.2   Command scripts

A script is a text file containing one or more BUA commands. A BUA command is an instruction to the Batch User Agent to perform an specific action, such as sending a message, listing the contents of folders in a subscriber's box, reading messages, and so on. BUA commands are tailored to the needs of applications and not optimized for human users. The following general rules apply to a BUA command script:

• Each command must be preceded by a dollar sign ($). For example:

```
$ LIST INFOBASE: FOLDER
$DELETE INFOBASE:PRIVATE
```

• A command may extend over more than one line. Only the first line of a command is preceded by a dollar sign ($). The dollar sign must be the first non-whitespace character (i.e. character other than a tab, a blank or end-of-line) in the line and the command itself must be in the same line as the dollar sign. E.g.

```
$ LIST
INFOBASE: FOLDER

    $DELETE INFOBASE:PRIVATE
```

• Spaces and tabs (but no newline characters) are permitted between a keyword and the separators ":" or "::".

• The maximum number of characters per line is 1024. Extra characters are truncated.

- Empty lines are allowed.

- Commands and tags are not case-sensitive. Filenames may or may not be case-sensitive, depending on the host system used.

- A tag/value notation is used to represent data values. Simple data values are entered as follows:

  tag_id : value

  Example
  ```
  SURNAME: PETERSON
  ```

- A value need not be on the same line as the associated keyword, i.e. the following input is correct:

  ```
  ENTRY_TYPE    :
            IPM
  ```

- Complex data values use a nested syntax:

  ```
  tag_id::sub_tag_id1 : value1 sub_tag_id2 ...(* can again be nested *)
  ... END
  ```

  Example
  ```
  ORG_UNIT_HIERARCHY:: ORG_UNIT: EIS ORG_UNIT: SIC END
  ```

  Each structure is introduced by a tag and a double colon and terminated by the keyword *END*. Every double colon must therefore be associated with a corresponding *END*. No whitespaces are allowed between the two colons in a double colon.

- The order of tag value pairs must be strictly observed, except where indicated to the contrary.

- Input values are trimmed, i.e. leading and trailing whitespaces are removed. In addition, whitespaces are compressed, i.e. a sequence of more than one whitespace is compressed to a single blank. Strings in double quotes (") are not trimmed and not compressed. Some address values are, however, always compressed (see table in appendix E on page 118). If a value itself contains double quotes, each double quote must be doubled. E.g., the input

  ```
  GIVENNAME:        Graham
  SUBJECT: "Some    queries on the ""X49"" project"
  ```

  will be interpreted as follows:

  ```
  GIVENNAME: Graham
  SUBJECT: "Some    queries on the ""X49"" project"
  ```

- String values can always be enclosed in double quotes. They <u>must</u> be enclosed in double quotes if they contain whitespaces.

- String values can extend over several lines up to a maximum logical line length of 1024 characters. The end-of-line character does not terminate a string.

- Each script must be terminated by an EXIT command. A missing EXIT command causes an error message ("Missing EXIT command").

  ```
  $LIST INFOBASE:FOLDER
  $DELETE INFOBASE:PRIVATE
  $EXIT
  ```

4.3   Result files

The Batch User Agent generates a result file each time it executes a command script. These result files are then stored in a result subdirectory.

The general structure and syntax of result files is described below. Detailed descriptions of the syntax used for the results of individual commands can be found in the relevant command descriptions in section 5.2.

- The original command that generated the result is shown in the result file. In the case of the *SUBMIT* command, the message part is omitted.

- Each line in the result file contains a single keyword and the relevant value, separated by a colon.

- Any error messages generated will also be written to the result file.

4.4   Journaling

For the event of errors, hardware or software failures during the processing of the input scripts BUA journaling gives the user the possibility to recover from the errors and to continue aborted handling of BUA files after the last completely processed script instruction.

If the processing of the OpenMS BUA process is aborted or not led correctly to end due to a system error (not syntax errors) or for lack of resources etc., the renewed call of OMS_BUA can restart the BUA script at the aborted position.

Restarting at the aborted instruction occurs automatically, if necessary with several attempts. After a pre-defined number of attempts (OMS_POLLER_DAT:OMS_POLLER_CNF.TXT; parameter „max_process_retry "; 0=unlimited) no more attempts is undertaken however the BUA file is placed into the operator's BAD directory.

4.5   Host access

The way in which the subscriber accesses the Batch User Agent varies according to the file transfer utility used. Three file transfer utilities are available with OpenMS:

- BSC-oriented file transfer protocol utility

- FTAM

- FTP

The following sections describe these three utilities and the associated login procedures in detail.

Note

Also OpenMS offers three file transfer utilities your X.400 mail service may not provide them all for communication with host server (for example BusinessMail X.400 only provides FTP

4.5.1   BSC access

BSC access has been implemented in OpenMS and allows you to access the system in the same way as with an IBM 2780/3780 station. An appropriate emulator must be installed on the host.

BSC access provides the following functions:

- Controlled login

  The system checks that the login is correct and a status file is returned to the host system.

- Sending of files

  Files to be sent contain BUA command scripts, text, EDI Transmission Sets and attachments.

- Receiving of files

  Files to be received contain the results of a BUA operation, text, EDI Transmission Sets and attachments.

- File support

  Two file formats are supported: variable record length format and undefined (stream) format. Refer to "Appendix F: File formats" on page 123 for further details of the variable and undefined file formats. Conversion between these formats is performed in some instances when files are transferred between the host system and the remote system. The table below shows when conversion is performed.

BUA logical directory	Filenames	Format required by OpenMS	Conversion performed
BUA_LOGIN_DIR	*.bua	variable	yes
BUA_LOGIN_DIR	*.ets	undefined	no
BUA_LOGIN_DIR	*.txt	undefined	yes
BUA_LOGIN_DIR	*.txt_att	undefined	yes
BUA_LOGIN_DIR	*.bin_att	undefined	no
BUA_RESULT_DIR	*.res	variable	yes
BUA_RTS_DIR	*.rts	undefined	no
BUA_TXT_DIR	*.<ddd>	undefined	yes
BUA_TXT_ATT_DIR	*.<ddd>	undefined	yes
BUA_FOR_ATT_DIR	*.<ddd>	variable	yes
BUA_BIN_ATT_DIR	*.<ddd>	undefined	no

Table 2: BUA logical directories, file formats and conversion

General procedure

The command syntax shown in this section may vary in places, depending on the file transfer products installed on different host systems.

Typically, BSC access to the Batch User Agent involves the following steps:

- Create the following files:

  - the login file for logging into OpenMS

  - the BUA command script

  - any text or binary attachments and EDI Interchanges required

  - a command file to control file transfer.

- Transfer all the files BUA will require when processing the command script. This must be done before the command script itself is transferred.

- Check the login status as returned in the status file. (This step is optional.)

- Transfer the BUA command script.

- Wait an appropriate length of time until the BUA has processed the request.

- Transfer the result file and any additional files (attachments etc.).

Note

The files must be sent in transparent mode.

The following example shows the steps that must be carried out on the host system if you wish to submit a message with the binary attachment "tar.bin_att". It is assumed that "tar.bin_att" has already been created. The names of the files created are shown in bold followed by the contents of the files.

1. Create the login file with the SEND option:

   SUBMIT_ATT.LOGIN

   ```
   LOGIN username password SEND
   ```

2. Create a command procedure to allow the BUA command script to be sent:

   SUBMIT_ATT_TRANSFER_SCRIPT.COM

   ```
   SEND SUBMIT.BUA
   ```

3. Create a command procedure to allow the binary attachment to be sent:

   SUBMIT_ATT_TRANSFER_MORE.COM

   ```
   SEND TAR.BIN_ATT
   ```

4. Create the LOGOUT command procedure:

   SUBMIT_ATT_LOGOUT.COM

   ```
   LOGOUT
   ```

5. Create the BUA command script:

SUBMIT_ATT.BUA

This example does not show the entire command script

```
$SUBMIT INFOBASE: SENTFOLDER
        IMPDU::
              IMPDU_HEADING::
                      PRIORITY: URGENT
                      ORIGINATOR::
                      ...
                        ...                ! Text omitted here
                      ...
              IMPDU_BODY::
                      TEXT:
                            This is the first line
                            This is the second line
                      EOTEXT
                      ATTACH::
                            TYPE: BIN
                            FILE: TAR.BIN_ATT
                      END
              END
        END
$EXIT
```

6. Create a command procedure to perform the entire BUA session:

SUBMIT_ATT.COM

```
call the file transfer utility
set up the line

send SUBMIT_ATT.LOGIN
receive SUBMIT_ATT.STATUS (check status)
send SUBMIT_ATT_TRANSFER_MORE.COM
receive SUBMIT_ATT.STATUS (check status)
send TAR.BIN
receive SUBMIT_ATT.STATUS (check status)
send SUBMIT_ATT_TRANSFER_SCRIPT.COM
receive SUBMIT_ATT.STATUS (check status)
send SUBMIT.BUA
send SUBMIT_ATT_LOGOUT.COM
```

7. You receive the result file in a similar manner.

Certain conventions must be adhered to when sending and receiving files under BSC. These are described in the rest of this section.

Sending single files

The following sequence must be observed when sending single files:

Send login file:	LOGIN USERNAME PASSWORD SEND
Receive action file:	LOGIN_SUCCESSFUL or LOGIN_FAILED
Send action file:	SEND file1.xxx
Receive action file:	FILENAME_RECEIVED: file1.xxx
Send file:	file1.xxx
Receive action file:	FILE_RECEIVED: file1.xxx
Send action file:	SEND file2.xxx
Receive action file:	FILENAME_RECEIVED: file2.xxx
Send file:	file2.xxx
Receive action file:	FILE_RECEIVED: file2.xxx
Send action or login file:	LOGOUT or LOGIN...

Notes

1. The only valid actions are LOGIN, SEND and LOGOUT.

2. The session is terminated when the subscriber sends LOGOUT.

3. If the action is invalid (an action other than LOGIN, SEND or LOGOUT or which does not observe the correct sequence), the user receives an action file containing "UNKNOWN_COMMAND_OR_INVALID_ACTION", and the session is terminated.

4. A timeout mechanism automatically terminates the session if the user does not send LOGIN, SEND or LOGOUT within a period of 1 minute.

5. Part of the file specification of the action record "SEND file.xxx" can be the BUA login directory specification, e.g. "bua_login_dir:".

Receiving single files:

The following sequence must be observed when receiving single files:

Send login file:	LOGIN USERNAME PASSWORD RECEIVE
Receive action file:	LOGIN_SUCCESSFUL or LOGIN_FAILED
Send action file:	RECEIVE file1.xxx
Receive file:	file1.xxx or FILE_NOT_ FOUND
Send action file:	RECEIVE file1.xxx
Receive file:	file1.xxx or FILE_NOT_ FOUND
Send action or login file:	LOGOUT or LOGIN...

Notes

1. The only valid actions are LOGIN, RECEIVE and LOGOUT.

2. The session is terminated when the subscriber sends LOGOUT.

3. If the action is invalid (an action other than LOGIN, RECEIVE or LOGOUT or which does not observe the correct sequence), the user receives an action file containing "UNKNOWN_COMMAND_OR_INVALID_ACTION", and the session is terminated.

4. A timeout mechanism automatically terminates the session if the user does not sent LOGIN, RECEIVE or LOGOUT within a period of 1 minute.

5. If the file does not exist, the user receives an action file containing "FILE_NOT_FOUND", and the session is terminated.

6. Part of the file specification of the action record "RECEIVE file.xxx" can be a legal BUA directory specification, e.g. "bua_text_dir:". If the directory is not specified, all user directories will be searched for the file.

Receiving a group of files:

The following sequence must be observed when receiving groups of files with the same name and different extensions (e.g. file.RES, file.001 : file.002 : file.nnn):

Send login file:	LOGIN USERNAME PASSWORD RECEIVE file.*
Receive action file:	LOGIN_SUCCESSFUL or LOGIN_FAILED
Send action file:	RECEIVE_FILENAME
Receive action file:	NEXT_FILE_IS: file.xxx
Send action file:	RECEIVE_FILE_DATA
Receive file:	file.xxx
Send action file:	RECEIVE_FILENAME
Receive action file:	NEXT_FILE_IS: file.yyy
Send action file:	RECEIVE_FILE_DATA
Receive file:	file.yyy
Send action file:	RECEIVE_FILENAME
Receive action file:	END
Send action or login file:	LOGOUT or LOGIN...

Notes

1. The file specification must be "*filename.**".

2. The first file that the user receives will be the file with the file extension ".RES". Next the user will receive all files in ascending order of filename extension, starting with ".001". Each file will be preceded by an action file with the following format: "NEXT_FILE IS: *directory:filename.nnn*". The "*directory:*" specification corresponds to one of the BUA logical directories (see Table 2 on page 22).

3. After all files have been transferred, the BSC user will receive an action file containing "END". The user can then send a new login file (LOGIN ...) or an action file containing "LOGOUT". A timeout mechanism automatically terminates the session if the user does not send LOGIN, SEND or LOGOUT within a period of 1 minute.

4. If the action is invalid (an action other than LOGIN, RECEIVE_FILENAME, RECEIVE_FILE_DATA or LOGOUT or which does not observe the correct sequence), the user receives an action file containing "UNKNOWN_COMMAND_OR_INVALID_ACTION", and the session is terminated.

Session termination:

The session will be terminated under the following conditions when sending or receiving files:

- if the user sends LOGOUT

- if the user sends an invalid login file or action file

- if an unrecoverable error occurs during a file transfer.

When using a switched line (dial in), the line will be disconnected and the user must dial in again and send "LOGIN" to start another session. On a non-switched line the line will not be dropped, and another session can be started by sending "LOGIN".

EBCDIC character set

The BSC system uses the login to determine whether the subscriber is using the ASCII or EBCDIC character set. If the subscriber is using EBCDIC, certain files will be translated from EBCDIC to ASCII and vice versa.

During a SEND operation, files with the following extensions will be translated from EBCDIC to ASCII:

- ".bua"

- ".txt"

- ".ebc"

- ".txt_att"

Files with other extensions will not be translated.

During a RECEIVE operation, files with the following extensions from the specified directories will be translated from ASCII to EBCDIC:

- ".res" in bua_result_dir

- ".ebc" in bua_rts_dir

- All files in bua_txt_dir

- All files in bua_txt_att_dir

- All files in bua_for_att_dir

In all other cases, translation will not take place.

Login file

The login file must contain only one line of text and must have the following structure:

`LOGIN username password action [file]`

The components of the LOGIN command have the following meaning:

LOGIN	Identifies the record as the login record. The keyword LOGIN must be followed by one space.
username	The user name. This specification must be followed by one space.
password	The password. This specification must be followed by 1 space.
action	Must be SEND for sending a file or RECEIVE for receiving files. This specification must be followed by one space if the login record includes a file specification.
file	File specification, filename (e.g. SUBMIT.*). If a file is specified, it is assumed that the user wants to receive a group of files. The syntax is "FILENAME.*". If the user wants to send or receive single files, the login record must not include a file specification.

The BSC system returns the following status information after it has received the login record:

- LOGIN_SUCCESSFUL or

- LOGIN_FAILED (syntax error, invalid user name, password or action)

Action file

The action file must contain only one line of text and must have the following structure:

- SEND file.xxx

- RECEIVE file.xxx

- RECEIVE_FILENAME

- RECEIVE_FILE_DATA

- LOGOUT

The components of the action file have the following meaning:

SEND file	Indicates that the user wants to send a file. SEND must be followed by one space and then the file specification. The file specification may contain the login directory name (e.g. "bua_logindir:"). Regardless of the directory specification, all files will be moved to the user's BUA_LOGIN_DIR.
RECEIVE file	Indicates that the user wants to receive a file. RECEIVE must be followed by one space and then the file specification. The file specification may contain a directory name (logical BUA result directory, e.g. BUA_TXT_DIR). If no directory name is specified, the system searches for the file in all user directories.
RECEIVE_FILENAME	Indicates that the user wants to receive the next available "filename" when receiving a group of files.
RECEIVE_FILE_DATA	Indicates that the user wants to receive the next available file ("data") when receiving a group of files.

LOGOUT   Indicates that the session is to be terminated. DTR will be dropped, thus disconnecting a switched line. On a non-switched line the BSC system waits for a new login record.

The BSC system returns the following status information after it has received an action record or data:

- FILENAME_RECEIVED: file.xxx (after user has sent "SEND file.xxx")

- FILE_RECEIVED: file.xxx (after user has sent "data" of file.xxx)

- FILE_NOT_FOUND (after user has sent "RECEIVE file.xxx" and file does not exist)

- NEXT_FILE_IS: file.xxx (after user has sent "RECEIVE_FILENAME")

- END (after user has sent "RECEIVE_FILENAME" and no more files of a group are available)

- UNKNOWN_COMMAND_OR_INVALID_ACTION (illegal user action or action is out of sequence).

4.5.2   FTAM access

FTAM is the OSI-based File Transfer, Access and Management method. Both the system administrator of the service-provider's system and the system administrator of the host system will have to set up address information before it is possible to access the OpenMS system. Please refer to the FTAM documentation for the host system for details.

The command syntax to be used for setting up addresses and carrying out file transfer varies from system to system and is described in the relevant FTAM documentation.

In order to minimize administration overhead, it is recommended that the OpenMS account name and password should be the same as the system account name and password.

FTAM uses the system account name and password for login rather than the OpenMS account name and password.

Under FTAM, the login procedure is carried out implicitly when a file is copied. No explicit login is required. The system administrator of the host system may have to provide the addressing information in separate steps before files are transferred. Again, this varies from system to system.

FTAM supports the following document types:

− FTAM-1 for unstructured text files

− FTAM-2 for sequential text files

− FTAM-3 for unstructured binary files.

The Batch User Agent only uses the FTAM-2 and FTAM-3 types as shown in the table below. Refer to "Appendix F: File formats" on page 123 for further details of the variable and undefined file formats.

BUA logical directory	Filenames	Format required by OpenMS	FTAM document type
BUA_LOGIN_DIR	*.bua	variable	FTAM-2
BUA_LOGIN_DIR	*.ets	undefined	FTAM-3
BUA_LOGIN_DIR	*.txt	undefined	FTAM-3
BUA_LOGIN_DIR	*.txt_att	undefined	FTAM-3
BUA_LOGIN_DIR	*.bin_att	undefined	FTAM-3
BUA_RESULT_DIR	*.res	variable	FTAM-2
BUA_RTS_DIR	*.rts	undefined	FTAM-3
BUA_TXT_DIR	*.<ddd>	undefined	FTAM-3
BUA_TXT_ATT_DIR	*.<ddd>	undefined	FTAM-3
BUA_FOR_ATT_DIR	*.<ddd>	variable	FTAM-2
BUA_BIN_ATT_DIR	*.<ddd>	undefined	FTAM-3

Table 3: BUA logical directories, file formats and FTAM document types

4.5.3   FTP access

The File Transfer Protocol (FTP) is available for many operating systems and might be used to transfer files to and from the OpenMS access host. ASCII and binary modes are supported as shown in the following table. Refer to "Appendix F: File formats" on page 123 for further details of the variable and undefined file formats.

BUA logical directory	Filenames	Format required by OpenMS	Transfer mode
BUA_LOGIN_DIR	*.bua	variable	ASCII
BUA_LOGIN_DIR	*.ets	undefined	Binary
BUA_LOGIN_DIR	*.txt	undefined	Binary
BUA_LOGIN_DIR	*.txt_att	undefined	Binary
BUA_LOGIN_DIR	*.bin_att	undefined	Binary
BUA_RESULT_DIR	*.res	variable	ASCII
BUA_RTS_DIR	*.rts	undefined	Binary
BUA_TXT_DIR	*.<ddd>	undefined	Binary
BUA_TXT_ATT_DIR	*.<ddd>	undefined	Binary
BUA_FOR_ATT_DIR	*.<ddd>	variable	ASCII
BUA_BIN_ATT_DIR	*.<ddd>	undefined	Binary

Table 4: BUA logical directories, file formats and transfer modes

FTP stores data in the form of streams and linefeeds (LF). Files transferred from OpenMS in ASCII mode will be stored with a linefeed. When transferred in binary mode, the End Of Record marker will not be translated into a linefeed.

An FTP session involves the following steps:

• Create the BUA command script and any attachments or EDI Interchanges.

• Start the FTP session.

• Log into the OpenMS access host.

• Send text files, attachments or Interchanges. These files must be transferred <u>before sending</u> the BUA script file.

• Send the BUA script file.

• Wait for the result file.

• Transfer the result file and any attachments or Interchanges to own server.

• Terminate the FTP session.

The name of the OpenMS access host, which is needed for login, can be obtained from the service provider together with the relevant user name and password.

The following example shows a session in which all files are retrieved from the SENTFOLDER. The numbers displayed in the example are implementation-dependent and can be ignored.

1. Create the BUA command script "fetch bua":

   ```
   $FETCH INFOBASE:SENTFOLDER $EXIT
   ```

2. Establish an FTP connection to the OpenMS access host:

   ```
   FTP host_name
   Name (host_name:username): username
   331 Password required for username.
   Password:
   230 User username logged in.
   ```

3. Check the directory name on the OpenMS computer to see whether it matches the subscriber's PBID:

   ```
   ftp>pwd
   257 "/bua/001000030.dir" is current directory.
   ```

4. Change mode to ASCII (default):

   ```
   ftp>ascii 200 Type set to A.
   ```

5. Send the BUA input script:

   ```
   ftp>send fetch.bua
   200 PORT command successful.
   150 Opening data connection for fetch.bua (16.185.192.203,1062).
   226 Transfer complete.
   local: fetch.bua remote: fetch.bua
   60 bytes sent in 0.031 seconds (1.9 Kbytes/s)
   ```

6. Terminate the FTP session:

   ```
   ftp>bye 221 Goodbye.
   ```

7. Wait for the result file to be transferred to the host system

8. Login as shown above.

9. Find the result file directory

   ```
   ftp> ls -l
   200 PORT command successful.
   150 Opening data connection for /bin/ls (16.185.192.203,1065) (0 bytes).
   total 9
   drwx------2 buatwo512 Nov 25 07:52 bin_att.dir
   drwx------2 buatwo512 Nov 25 07:52 for_att.dir
   drwx------2 buatwo512 Nov 25 07:52 result.dir
   drwxr-xr-x2 buatwo512 Nov 25 07:52 rts.dir
   drwx------2 buatwo512 Nov 25 07:52 txt.dir
   drwx------2 buatwo512 Nov 25 07:52 txt_att.dir
   226 Transfer complete.
   remote: -l
   348 bytes received in 0.27 seconds (1.3 Kbytes/s)
   ```

10. Change to the result file directory:

   ```
   ftp> cd result
   250 CWD command successful.
   ftp> pwd
   257 "/bua/001000030.dir/result.dir" is current directory.
   ```

11. Check whether result file is present:

```
ftp> ls -l
200 PORT command successful.
150 Opening data connection for /bin/ls (16.185.192.203,1066) (0 bytes).
total 1
-rwxr-x---1 buatwo          536 Nov 23 11:28
fetch.res.1
226 Transfer complete.
remote: -l
180 bytes received in 0.23 seconds (0.78 Kbytes/s) ftp>
```

12. Fetch the result file in ASCII mode:

```
ftp> ascii
200 Type set to A.
ftp> get fetch.res
200 PORT command successful.
150 Opening data connection for fetch.res (16.185.192.203,1067) (536 bytes).
226 Transfer complete.
local: fetch.res remote: fetch.res
517 bytes received in 0.066 seconds (7.6 Kbytes/s)
```

13. Either look for more files (e.g. attachments) or disconnect from OpenMS:

```
ftp>bye
221 Goodbye.
```

14. Check that the result file is on your system:

```
local_prompt ls -l
total nnn
-rw-r--r--1 local_username          536 Nov 25 11:04 fetch.res
```

4.6   **Direct Batch User Agent**

There are customers, who only send attachments. In order not to process these attachments through the whole OpenMS environment thus to avoid potential problems (long run times of processes, like MTA, DLV, BUA - Submit and - Fetch), the so-called „Direct BUA" is provided. The users of Direct BUA must be aware the restriction that this service is **not** a real X.400 service.

Direct BUA is a service, similar to FTP, which delivers mail attachments of a customer directly into the recipient's BUA directory. Thereupon the OpenMS BUA process creates a pseudo result file in the recipient directory. This contains among others, information about senders, recipients, time-of-day, subject and name of the delivered file. If the sender requires, he can receive „Delivery Reports", but using the mechanisms of Direct BUA it is not possible to create Read Reports or Receipt Notifications.

The BUA process checks before executing a COPY of the data to a directBUA-User, if the respective directBUA User directories are present on the same device as the source file. If this is the case then a RENAME instead of a COPY will be executed.

As soon as the OpenMS BUA fetched the attachment from the sender's directory, the sender is not able to access this file any more. As soon as the recipient deleted the attachments from his BUA directory, these are lost irreparably.

Using Direct BUA it is not possible to reach receivers who do not access their BUA directories via FTP. It is also not possible to address recipients in this way outside of the OpenMS boundaries. Also receivers are excluded who are connected to OpenMS over P7 or LUA.

Direct BUA is applicable only for certain senders with defined recipients. The OpenMS operator defines the possible Direct BUA partnerships (OMS_POLLER_DAT:OMS_POLLER_HOST_PROFILE. DAT, parameter "DIRECT_BUA").

```
MPDUID-format, which will be found in the BUA-result file:

MPDUID := DirectBUA<YYYYMMTTHHMMSS><Counter><User-ID>

<Counter> is a 2-digit hexadecimal number 00-FF
<User-ID> is a 6-digit hexadecimal number of the
originator's user_ID
```

5   Command descriptions

This chapter is divided into two parts. The first part, section 5.1, *Introduction*, provides a general introduction to the command set. The second part, section 5.2, *Command reference*, consists of in-depth descriptions of the available commands in alphabetical order.

5.1   Introduction

This section contains an overview of the available commands in functional groups, general descriptions of the command syntax, and a guide to the notational conventions used in the command descriptions.

5.1.1   Command overview

The following table contains all the commands in the Batch User Agent command set, with a brief synopsis of its characteristics and page references to the full descriptions of the listed commands.

Command	Mailbox for which command is available	Function	Refer to:
CONTROL	Both	Controls whether or not commands in a script are executed in the event of an error	page 39
DELETE	Both	Deletes messages from a selected infobase	page 40
DOWNLOAD	EDI	Writes an EDI Transmission Set to the subscriber's directory	page 42
EXIT	Both	Terminates the session	page 44
FETCH	Both	Fetches selected messages and writes the headers to the result file and the body parts and attachments to separate files	page 45
FILE	MAIL	Moves messages from one folder to another	page 63
LIST FOLDER	Both	Returns information on folders and contents	page 65
LIST <message_folder>	Both	Returns information on the contents of specified folders	page 67
LIST PROFILE	EDI	Returns information on the current user profile	page 70
LIST SUBSCRIBER	MAIL	Returns information on subscribers	page 71
MODIFY	MAIL	Modifies user folder names	page 73
REGISTER	EDI	Changes the number of days for which EDI documents are archived	page 75
REMARK	Both	Inserts comments in a command script	page 77
(selector)	Both	Description of the message selector available with many of the commands	page 78

Command	Mailbox for which command is available	Function	Refer to:
STATUS	EDI	Generates an EDI activity report	page 82
STORAGE	Both	Returns the number of storage units currently used	page 85
SUBMIT	MAIL	Creates and sends a message	page 87
UPLOAD	EDI	Transfers an EDI Transmission Set from the subscriber's directory to the subscriber's EDI folder	page 100

Table 5: Command overview

5.1.2 Command syntax

The syntax of Batch User Agent commands can be generalized in the following form:

```
$VERB INFOBASE SELECTOR QUALIFIER
```

Some commands, however, have no qualifier, while others do not use the selector and yet others consist of the verb only.

The *$verb* describes the action you wish to perform. Section 5.2 lists commands alphabetically in order of verb.

The *infobase* describes what the verb is to act on (e.g. FOLDER, PROFILE etc.) or which folder to use (e.g. INFOLDER, SENTFOLDER etc.).

The *selector* defines a particular object or objects in the selected infobase. Thus if there are five messages numbered 1 through 5, *SEQ_NR:2* selects message number 2.

The *qualifier* selects a subset of the information identified by the previous elements of the command or switches a state (e.g. ON/OFF).

5.1.3 Infobases

The following infobases are defined for **MAIL box** subscribers:

- **SENTFOLDER**: contains all messages with entry status SENT.
- **INFOLDER**: contains received messages with entry status NEW, LISTED or OLD.
- **User-defined folders**: contain messages.
- **FOLDER**: contains folder objects.
- **SUBSCRIBER**: contains subscriber objects.

The following infobases are defined for **EDI box** subscribers:

- **SENTFOLDER**: contains all EDI documents with entry status SENT.
- **INFOLDER**: contains received EDI documents with entry status NEW, LISTED or OLD.
- **FOLDER**: contains folder objects.

- **MAILBOX**: contains the various kinds of reports.

- **PROFILE**: contains the local profile.

Infobases and the Local User Agent

When subscribers access an account through the Local User Agent they see a different structure for the system-defined folders. The two folder structures correspond to each other as follows:

Batch User Agent	Local User Agent (BUA: MAIL box user)	Local User Agent (BUA: EDI box user)
INFOLDER	MAILBOX	EDIINFOLDER
SENTFOLDER	SENTFOLDER	EDIOUTFOLDER
MAILBOX	N/A	MAILBOX

Table 6: Folder structure in the Batch User Agent and the Local User Agent

Note

You cannot access the Batch User Agent when you are already logged in to the Local User Agent or vice versa.

5.1.4   Other conventions

Case sensitivity

Almost all verbs (*SUBMIT*, *LIST*) and objects (*INFOBASE*, *FOLDER*) and all keywords for predefined selectors (*CURRENT*, *STATUS*) and qualifiers (*ON, FILE*) are shown in uppercase and in full form. These tokens are all case-insensitive: you can type them in uppercase or lowercase or a mixture of the two.

Table of notational conventions

Notation	Meaning
[square brackets]	Square brackets enclose optional elements. You do not type the square brackets.
<angle brackets>	A string in angle brackets stands for a value, which you can or must specify. You do not type the angle brackets.
{braces|vertical bar}	Braces enclose lists of mutually exclusive alternatives separated by vertical bars. You do not type the braces or the vertical bars.
...	An ellipsis indicates that the previous item can be repeated any number of times.
underscored default	In lists of alternatives, the underscored value is the default.

Table 7: Notational conventions

Matching rules

As described above, all keywords are case-insensitive. Variable strings in the command script, such as attribute values, are subject to the following matching rules:

- String matching is case-insensitive, even if the input string is quoted. Thus a folder name can be typed either in uppercase or lowercase or a mixture of the two and may be quoted or not.

- Matching is always exact with the exception of selectors which use the keyword *CONTAINS*.

5.2   Command reference

This section is an alphabetically arranged reference to the Batch User Agent command set. Not all commands are available to all users. This is indicated at the appropriate points.

One of the entries included in this reference, *selector*, is not strictly speaking a user command. The use of lowercase for it is indicative of its special status.

selector is included as it is a complex parameter common to many commands and is therefore described only once. The relevant commands all include a cross-reference to this entry.

CONTROL

This command controls whether or not commands in a script are executed in the event of an error. Box type: EDI box and MAIL box.

Syntax

$CONTROL {ON|OFF}

Parameters

ON   specifies that all subsequent commands are to be ignored if a syntax error is detected in the command file.

OFF   specifies that any subsequent commands are processed after a syntax error is detected in the command file.

Description

The *CONTROL* command is used to control the execution of commands in a script. The two possible parameters are *ON* or *OFF*. If *ON* is set and a syntax error is detected in the command file, execution of the script is aborted after the failed command. If *OFF* is set, execution of the script continues with the next command.

The default setting immediately after login is *ON*.

Result syntax

```
RESULT::
  CONTROL_IS_SET:{ON|OFF}
END
```

Result data

CONTROL_IS_SET:{ON|OFF}
                         indicates the setting after the *CONTROL* command has been issued.

Example

Contents of command script
```
$CONTROL OFF
```

Contents of result file
```
COMMAND: CONTROL OFF
RESULT::
CONTROL_IS_SET: OFF
END
```

DELETE

This command deletes messages.

Box type: EDI box and MAIL box.

Syntax

$DELETE INFOBASE:<infobase_name> [*selector*]

Parameters

INFOBASE:<infobase_name>

>> *infobase_name* can contain one of the following values: *SENTFOLDER*, *INFOLDER*, *MAILBOX* (EDI box only), *<user_defined_folder>* (MAIL box only). Refer to section 3.4 on page 16 for details of these specifications.

selector   The selector identifies the messages, which are to be deleted. The available selectors and the message selector syntax are described on page 78.

>> Default: All messages in the selected infobase.

Description

The *DELETE* command is used to remove one or more messages from an infobase. Messages that you delete are discarded immediately and irretrievably. There is no wastebasket and no *UNDELETE* command. Messages with the status *NEW* cannot be deleted.

Note:

After a DELETE operation the keyword CURRENT is not defined if the "current" message has been deleted.

Result syntax

```
RESULT::
  NUMBER_DELETED_OBJECTS:<integer>
END
```

Result data

NUMBER_DELETED_OBJECTS:<integer>

>> *integer* returns the number of messages deleted.

Example

This example illustrates the use of the *DELETE* command with no qualifiers. All the messages in the selected infobase will be deleted.

Contents of command script
```
$DELETE INFOBASE:FN01
```

Contents of result file
```
COMMAND: DELETE INFOBASE:FN01
RESULT::
NUMBER_DELETED_OBJECTS: 7
END
```

DOWNLOAD

This command writes an EDI Transmission Set to the subscriber's directory.

Box type: EDI box.

Syntax

$DOWNLOAD [INFOBASE:<infobase_name>] [selector] FILE:<filename>

Parameters

INFOBASE:<infobase_name>

infobase_name can contain one of the following values: *SENTFOLDER*, *INFOLDER* (default). Refer to section 3.4 on page 16 for details of these specifications.

selector   The selector identifies the messages, which are to be downloaded. The available selectors and the message selector syntax are described on page 78.

Default: All messages with the entry status *NEW* or *LISTED* in the selected infobase.

FILE:<filename>   *filename* must specify the filename of a Transmission Set to be downloaded. The filename extension must be "*.rts"* (Result Transmission Set).

Description

The *DOWNLOAD* command is used to write an EDI document to the subscriber's directory. The command operates on documents with the entry status *NEW, LISTED, SENT* or *OLD*. When an EDI document with the status *NEW* or *LISTED* has been downloaded, its status is changed to *OLD*. The selected documents are copied from the EDI folder to BUA_RTS_DIR and stored in a single Transmission Set under the name "*filename*.rts". The Transmission Set can then be copied to the host system using an appropriate file transfer protocol.

Result syntax

```
RESULT::
  DOWNLOAD_FILE:<filename>
  DOWNLOAD_NUMBER:<integer>
  DOWNLOAD_STATUS:{TRANSFERRED|NO_NEW_ETS|FAILED}
END
```

Result data

DOWNLOAD_FILE:<filename>

filename indicates the name of the downloaded file.

DOWNLOAD_NUMBER:<integer>

integer returns the number of downloaded Interchanges

DOWNLOAD_STATUS:{TRANSFERRED|NO_NEW_ETS|FAILED}
indicates the *DOWNLOAD* command status.

TRANSFERRED
The download was successful.

NO_NEW_ETS
No new Transmission Sets were available in the location specified.

FAILED
The download was unsuccessful.

Example

This example demonstrates the use of the *TIME_RANGE* and *SUBJECT* selectors with the *DOWNLOAD* command. All EDI documents within the specified time range and whose subject contains the string "R24" are downloaded.

Contents of command script
```
$DOWNLOAD INFOBASE:INFOLDER TIME_RANGE::
                     AFTER:  19941201000000Z
                     BEFORE: 19950102000000Z
                 END
               SUBJECT CONTAINS "R24"
        FILE: R24.RTS
```

Contents of result file
```
COMMAND: DOWNLOAD INFOBASE:INFOLDER
                TIME_RANGE::
                     AFTER:  19941201000000Z
                     BEFORE: 19950102000000Z
                 END
               SUBJECT CONTAINS "R24"
        FILE: R24.RTS

RESULT::
DOWNLOAD_FILE: "BUA_RTS_DIR:r24.rts"
DOWNLOAD_NUMBER: 3
DOWNLOAD_STATUS: TRANSFERRED
END
```

EXIT

This command terminates the session. Box type: EDI box and MAIL box.

Syntax

$EXIT

Parameters

The *EXIT* command has no parameters.

Description

The *EXIT* command must appear as the last command in any command script. It serves to terminate the session and no further commands are executed after the *EXIT* command.

Result syntax

```
RESULT::
  NUMBER_NEW_MESSAGES:<integer>
  LOGOUT_BANNER::<string>
  END
END
```

Result data

NUMBER_NEW_MESSAGES:<integer>

                integer returns the number of new messages in all infobases.

LOGOUT_BANNER::<string> END

                string contains the logout banner displayed at the end of any OpenMS session.

Example

Contents of command script
```
$EXIT
```

Contents of result file
```
COMMAND: EXIT
RESULT::
NUMBER_NEW_MESSAGES: 0
LOGOUT_BANNER::

    Good Bye!

END
END
```

FETCH

This command fetches selected messages and writes the headers to the result file and the body parts and attachments to separate files.

Box type: EDI box and MAIL box.

Syntax

$FETCH INFOBASE:<infobase_name> [selector] [{HEADER|BOTH}]

Parameters

INFOBASE:<infobase_name>

> *infobase_name* can contain one of the following values:
> EDI box: *MAILBOX*
> MAIL box: *SENTFOLDER, INFOLDER,*
>   *<user_defined_folder>*.
> Refer to section 3.4 on page 16 for details of these specifications.

selector   The selector identifies the messages that are to be fetched. The available selectors and the message selector syntax are described on page 78.

> Default: All messages in the selected infobase.

HEADER   specifies that header data is written to the result file, but that no body part files are written to the relevant directories.

BOTH   (default). Specifies that header data is written to the result file and that body part files are written to the relevant directories. The header data in the result file shows the filename and the location of the body parts.

Description

The *FETCH* command gets the specified messages from the mailbox and writes the headers of all the messages, which have been fetched to the result file. If the qualifier *BOTH* is specified, body parts, attachments etc. are stored in the appropriate directories in accordance with the following table:

Logical directory name	Contents	Type
BUA_TXT_DIR	Text body parts	TXT
BUA_TXT_ATT_DIR	Text attachments, distribution lists, forms	TXT
BUA_BIN_ATT_DIR	Attachments in binary format	BIN
BUA_FOR_ATT_DIR	Forwarded messages	FOR

Table 8: Directories used by the FETCH command

The names and locations of the files are shown in the result file. The filename extensions for the text body parts and attachments consist of an alphanumeric string that is incremented for each file generated with the *FETCH* command. The counter is initialized to "001" at login time and consists of exactly three alphanumeric characters incremented according to the following pattern ("001" - "009", "00a" - "00z", "010" - "019", "01a" - "zzz").

The following example shows a list of files which have been placed in the TXT directory using two *FETCH* commands:
$FETCH INFOBASE:SENTFOLDER
$FETCH INFOBASE:INFOLDER
The corresponding BUA command files are fetch_in.bua and fetch_send.bua.

```
fetch_in.001

fetch_in.002

fetch_in.003

fetch_in.004

fetch_in.005

fetch_in.006

fetch_sent.001

fetch_sent.002

fetch_sent.003

fetch_sent.004

fetch_sent.005

fetch_sent.006

fetch_sent.007

fetch_sent.008
```

The status of a message that has been fetched changes as follows:

• Interpersonal messages with the status *NEW* or *LISTED* are assigned the status *OLD* if the qualifier *BOTH* is specified.

• Interpersonal messages with the status *NEW* are assigned the status *LISTED* if the qualifier *HEADER* is specified.

• Non-interpersonal messages (reports) are always assigned the status *OLD*.

Restriction

For EDI boxes, the *FETCH* command is only valid for reports (delivery reports, non-delivery reports, status reports and error reports).

If the limit for the maximum number of nested forwarded messages is exceeded (see Appendix B on page 105), no error message is generated, but the result file contains an attachment of the type *TXT* (instead of *FOR*) for the first bodypart, which exceeds this limit. This text attachment has the following contents:

```
" >>> Maximum number of nested bodyparts exceeded! <<< "
```

Result syntax

The result syntax for the *FETCH* command is complex, since it includes a number of nested substructures. It will be presented here in the form of a number of syntax diagrams, each with descriptions of the relevant output data.

```
RESULT::
  [fetched_object] ...
  NUMBER_FETCHED_OBJECTS:<integer>
END
```

Result data

fetched_object   See the syntax diagram below for a description of the syntax of this
                                structure. The structure is repeated for each message fetched by the
                                FETCH command.

NUMBER_FETCHED_OBJECTS:<integer>
                                integer returns the number of messages fetched by the *FETCH*
                                command.

Syntax of fetched_object

```
OBJECT::
FOLDER:<folder_name>
SEQ_NR:<sequence_no>
ENTRY_STATUS:{NEW|LISTED|OLD|SENT}
TIME:<time>
ENTRY_TYPE:<entry_type>
MESSAGE_SIZE:<integer>
HAS_ATTACHMENTS:{YES|NO}
entry_information
END
```

Result data for fetched_object

FOLDER:<folder_name>   *folder_name* indicates the name of the folder in which the message
                        is stored.

SEQ_NR:<sequence_no>   *sequence_no* is a 16-digit hexadecimal number indicating the
                        sequence number assigned to the message.

ENTRY_STATUS:{NEW|LISTED|OLD|SENT}
                        indicates the entry status of the message.

TIME:<time>   *time* indicates the delivery time of the message. The supported date
                        and time format is described in section 3.2 on page 14.

ENTRY_TYPE:<entry_type>
                        entry_type indicates the message type. Refer to appendix G on
                        page 124 for details of the possible values.

MESSAGE_SIZE:<integer>
                        integer returns the size of the message (without envelope) in bytes.

HAS_ATTACHMENTS:{YES|NO}
    indicates whether the message has any attachments or not.

> Note: If the first body part is the only body part and if this body part is a text body part, the value of *HAS_ATTACHMENTS* is *NO*.

entry_information   See the syntax diagrams below for a description of the syntax of this structure. The syntax of *entry_information* varies according to whether the message is a delivery report (DR), a status report (SR) or a message received from another subscriber (IMPDU). *entry_information* is empty for body parts of an unknown type.

Syntax of entry_information for delivery reports

```
DR::
  DRPDU::
    x400_address
    TIME:<time>
    MPDUID::
      COUNTRY: <country>
      ADMD: <admin domain>
      [PRMD: <private domain>]
      ID: <local id>
    END
    UA_CONT_ID:<uac_id>
    RECIPIENT_REPORT_INFO::
      REPORT_INFO::
        x400_address
        ARRIVAL_DATE:<arrival_time>
        {DELIVER_DATE:<delivery_time>
        |
        NON_DELIVERED_INFO::
          REASON:<integer>
          [DIAGNOSTIC:<integer>]
        END}
      END ...
    END
  END
END
```

Result data for entry_information for delivery reports

x400_address   See the syntax diagram following the message syntax diagram for details of the *x400_address* structure.

TIME:<time>   *time* indicates the arrival time of this delivery report (not of the original message). The supported date and time format is described in section 3.2 on page 14.

UA_CONT_ID:<uac_id>   *uac_id* contains the User Agent Content ID.

ARRIVAL_DATE:<arrival_time>

> *arrival_time* indicates the time that the original message arrived in the subscriber's message store. The supported date and time format is described in section 3.2 on page 14.

DELIVER_DATE:<delivery_time>

> *delivery_time* indicates the time that the original message was retrieved from the message store by the subscriber. The supported date and time format is described in section 3.2 on page 14.

REASON:<integer>

> *integer* is an 8-digit hexadecimal code indicating a reason for non-delivery. See Appendix D on page 114 for a description of the possible reason codes.

DIAGNOSTIC:<integer>

> *integer* is a 8-digit hexadecimal code providing diagnostic information in the event of non-delivery. See Appendix D on page 114 for a description of the possible diagnostic codes.

Syntax of entry_information for status reports

```
SR::
  RECEIPT_NOTIFICATION::
    [x400_address]
    IPM_ID::LOCAL_IPM_ID:<ipm_id> [x400_address] END
    RECEIPT:<time>
    TYPE_OF_RECEIPT:{EXPLICIT|AUTOMATIC}
  END
END
```

or

```
SR::
  NON_RECEIPT_NOTIFICATION::
    [x400_address]
    IPM_ID::LOCAL_IPM_ID:<ipm_id> [x400_address] END

NON_RECEIPT_REASON:{UAE_INITIATED_DISCARDED|AUTO_FORWARDED}
    [NON_RECEIPT_QUALIFIER:{EXPIRED|OBSOLETED|
     SUBSCRIPTION_TERMINATED|DELETED}]
  END
END
```

Result data for entry_information for status reports

x400_address

> See the syntax diagram following the message syntax diagram for details of the *x400_address* structure.

ipm_id

> contains the interpersonal message ID of the original message.

> Note: The X.400 address associated with this structure does not have a form tag.

RECEIPT:<time>

> *time* indicates the time the original message was received.

TYPE_OF_RECEIPT:{EXPLICIT|AUTOMATIC}

> indicates whether the receipt notification was generated as a result of a default setting or an explicit request.

NON_RECEIPT_REASON :{UAE_INITIATED_DISCARDED|AUTO_FORWARDED}
indicates the reason for Non-Receipt.
UAE_INITIATED_DISCARDED indicates that the IPM was
discarded. Further information may be contained in the
NON_RECEIPT_QUALIFIER field.
AUTOFORWARDED indicates that the IPM was not read by the
recipient but automatically forwarded by that recipient's Message
Store to another recipient.

NON_RECEIPT_QUALIFIER :{EXPIRED|OBSOLETED|
SUBSCRIPTION_TERMINATED|DELETED}
provides further information on the reason for Non-Receipt. The
possible values have the following meanings:

EXPIRED   The auto-discard mechanism was in effect and the
               expiry time had elapsed.

OBSOLETED
               The auto-discard mechanism was in effect and the
               message was identified as obsolete.

SUBSCRIPTION_TERMINATED
               The messaging system subscription of the intended
               recipient has been terminated.

DELETED   The recipient deleted the message without reading
               it.

Syntax of entry_information for interpersonal messages

The following syntax applies for an interpersonal message. The body information is only
returned if the qualifier *BOTH* or no qualifier is specified (BOTH is default) with the *FETCH*
command. The first tag (*HEADER* or *CONTENT*) indicates which qualifier has been selected.

```
{HEADER::|CONTENT::}
  IMPDU::
    IMPDU_HEADING::
      ENTRY_STATUS:{NEW|LISTED|OLD|SENT}
      TIME:<time>
      UA_CONT_ID:<uac_id>
      IPM_ID::LOCAL_IPM_ID:<ipm_id> [x400_address] END
      PRIORITY:{NONURGENT|NORMAL|URGENT}
      [IN_REPLY_TO::
        UA_CONT_ID:<ipm_id> [x400_address]
      END]
      ORIGINATOR::x400_address ... END
      [SUBJECT:<subject_string>]
      [EXPIRY_DATE:<expiry_date>]
      [REPLY_BY_DATE:<reply_by_date>]
      [SENSITIVITY:<sensitivity>]
      [IMPORTANCE:<importance>]
      [PRIMARY_RECIPIENTS::x400_address ... END]
      [COPY_RECIPIENTS::x400_address ... END]
      [BLIND_COPY_RECIPIENTS::x400_address ... END]
      [AUTHORIZING_USERS::x400_address ... END]
      [REPLY_TO_USERS::x400_address ... END]
      [CROSS_REFERENCE::
        IPM_ID::LOCAL_IPM_ID:<ipm_id> [x400_address] END
        ...
      END]
      [OBSOLETES::
        IPM_ID::LOCAL_IPM_ID:<ipm_id> [x400_address] END
        ...
      END]
    END
    [IMPDU_BODY::
      [TEXTFILE:<filename>]
      [ATTACH::
        TYPE:{TXT|BIN|FOR}
        FILE:<filename>
      END ...]
    END]
  END
END
```

The syntax for a forwarded message is as follows:

```
CONTENT::
  IMPDU::
    IMPDU_HEADING::
      DELIVER_TIME:<time>
      IPM_ID::LOCAL_IPM_ID:<ipm_id> [x400_address] END
      [IN_REPLY_TO::
        UA_CONT_ID:<ipm_id> [x400_address]
      END]
      ORIGINATOR::x400_address ... END
      [SUBJECT:<subject_string>]
      [EXPIRY_DATE:<expiry_date>]
      [REPLY_BY_DATE:<reply_by_date>]
      [SENSITIVITY:<sensitivity>]
      [IMPORTANCE:<importance>]
      [PRIMARY_RECIPIENTS::x400_address ... END]
      [COPY_RECIPIENTS::x400_address ... END]
      [BLIND_COPY_RECIPIENTS::x400_address ... END]
      [AUTHORIZING_USERS::x400_address ... END]
      [REPLY_TO_USERS::x400_address ... END]
      [CROSS_REFERENCE::
        IPM_ID::LOCAL_IPM_ID:<ipm_id> x400_address END
        ...
      END]
      [OBSOLETES::
        IPM_ID::LOCAL_IPM_ID:<ipm_id> x400_address END
        ...
      END]
    END
    [IMPDU_BODY::
      [ATTACH::
        TYPE:{TXT|BIN|FOR}
        FILE:<filename>
      END ...]
    END]
  END
END
```

Result data for entry_information for interpersonal messages

ENTRY_STATUS:{NEW|LISTED|OLD|SENT}
                 indicates the entry status of the message.

TIME:<time>   For messages in *SENTFOLDER*, *time* indicates the submission time of the message; for incoming messages, *time* indicates the delivery time of the message. The supported date and time format is described in section 3.2 on page 14.

DELIVER_TIME:<time>   *time* indicates the delivery time of a forwarded message. The supported date and time format is described in section 3.2 on page 14.

UA_CONT_ID:<uac_id>

                 uac_id contains the User Agent Content ID.

ipm_id   contains the interpersonal message ID of the original message.

                   Note: The X.400 address associated with this structure
                                    does not have a form tag.

PRIORITY:{NONURGENT|NORMAL|URGENT}
                   specifies the message priority.

IN_REPLY_TO::UA_CONT_ID:<ipm_id> [x400_address] END
                   If the message is a reply to a message you have sent previously,
                   ipm_id contains the interpersonal message ID of this message.

                   Note: The X.400 address associated with this structure
                                    does not have a form tag.

ORIGINATOR::x400_address END
                   This specification provides addressing information on the sender
                   (originator) of the message. The structure of *x400_address* is
                   described below.

SUBJECT:<subject_string>
                   subject_string contains a subject for the message. The subject can
                   be any explanatory string up to 128 characters long.

EXPIRY_DATE:<expiry_date>
                   expiry date indicates when the authorizing users consider the IPM
                   to lose its validity. The supported date and time format is described
                   in section 3.2 on page 14.

REPLY_BY_DATE:<reply_by_date>
                   expiry date indicates by when the authorizing users request (but do
                   not demand) that any replies to this IPM should be sent. The
                   supported date and time format is described in section 3.2 on page
                   14.

SENSITIVITY:<sensitivity>
                   sensitivity indicates the sensitivity that the authorizing users
                   attribute to the IPM. The sensitivity is expressed as a hexadecimal
                   number, which can take one of three values:

                   1   personal

                   2   private

                   3   company-confidential

IMPORTANCE:<importance>
                   importance indicates the importance that the authorizing users
                   attribute to the IPM. The importance is expressed as a hexadecimal
                   number that can take one of three values:

                   0   low

                   1   normal

                   2   high

                   The users themselves define the precise meaning of the importance
                   specification.

PRIMARY_RECIPIENTS::x400_address END

      This specification contains the name and address of the primary recipient or recipients. The structure of *x400_address* is described below.

COPY_RECIPIENTS::x400_address END

      This specification contains the names and addresses of carbon-copy recipients. The structure of *x400_address* is described below.

BLIND_COPY_RECIPIENTS::x400_address END

      This specification contains the names and addresses of blind-copy recipients. The structure of *x400_address* is described below.

      A blind copy recipient has the same role as a copy recipient, except that the blind copy recipient is intended to remain unknown to primary recipients and other copy recipients.

AUTHORIZING_USERS::x400_address END

      This specification contains the names and addresses of any authorizing users. The structure of *x400_address* is described below.

      An authorizing user is a user who, either individually or jointly with others, authorizes the origination of an IPM. This specification is present only if the authorizing users are other than the IPM's originator alone.

      Suppose, for example, that a manager instructs his or her secretary to originate an IPM on his or her behalf. In this case, the secretary, the IPM's originator, might consider the manager to be the authorizing user.

REPLY_TO_USERS::x400_address END

      This specification contains the names and addresses of preferred recipients of any reply to this message as specified by the originator. The structure of *x400_address* is described below.

CROSS_REFERENCE::IPM_ID::LOCAL_IPM_ID:<ipm_id> [x400_address] END ... END

      ipm_id identifies IPMs that the authorizing users of the present IPM consider related to it. The structure of *x400_address* is described below.

OBSOLETES::IPM_ID::LOCAL_IPM_ID:<ipm_id> [x400_address] END ... END

      ipm_id identifies IPMs that the authorizing users of the current IPM consider to be rendered obsolete by the current IPM. The structure of *x400_address* is described below.

TEXTFILE:<filename>   The text of a message is stored in a file. *filename* specifies the name of the file containing the text.

ATTACH::TYPE:{TXT|BIN|FOR} FILE:<filename> END

      specifies any attachments to the message. An attachment is a copy of another message or a file (typically a graphics file, spreadsheet etc.) appended to the message. Three values are possible for the type: *TXT* (text file), *BIN* (binary file) or *FOR* (forwarded message). *filename* specifies the file in which the attachments are contained.

The syntax of *x400_address* can take three different forms. The three formats are similar in content and the specifications are described together following the three syntax diagrams.

Form 1:

```
X400F1::
    COUNTRY:<country>
    ADMD:<admd>
    [PRMD:<prmd>]
    {[COMMON_NAME:<common_name>]   (×)
     |
     [PERSONAL_NAME::
         SURNAME:<surname>
         [GIVENNAME:<givenname>]
         [INITIALS:<initials>]
         [GENERATION:<generation>]
      END]}
    [ORGANIZATION:<orgname>]
    [ORG_UNIT_HIERARCHY::
        ORG_UNIT:<org_unit> [ORG_UNIT:<org_unit>]
        [ORG_UNIT:<org_unit>] [ORG_UNIT:<org_unit>]
    END]
    [FREEFORMNAME:<freeform_name>]
    [TELEPHONENUMBER:<telephone_number>]
    [DDALIST::
        DDA::DDA_TYPE:<ddatype> DDA_VALUE:<ddavalue> END
        [DDA::DDA_TYPE:<ddatype> DDA_VALUE:<ddavalue> END]
        [DDA::DDA_TYPE:<ddatype> DDA_VALUE:<ddavalue> END]
        [DDA::DDA_TYPE:<ddatype> DDA_VALUE:<ddavalue> END]
    END]
    [RECEIPT_NOTIFICATION:{YES|NO}]
    [NON_RECEIPT_NOTIFICATION:{YES|NO}]
END
```

Form 2:

```
X400F2::
    COUNTRY:<country>
    ADMD:<admd>
    [PRMD:<prmd>]
    UA_CONT_ID:<uaid>
    [FREEFORMNAME:<freeform_name>]
    [TELEPHONENUMBER:<telephone_number>]
    [DDALIST::
        DDA::DDA_TYPE:<ddatype> DDA_VALUE:<ddavalue> END
        [DDA::DDA_TYPE:<ddatype> DDA_VALUE:<ddavalue> END]
        [DDA::DDA_TYPE:<ddatype> DDA_VALUE:<ddavalue> END]
        [DDA::DDA_TYPE:<ddatype> DDA_VALUE:<ddavalue> END]
    END]
    [RECEIPT_NOTIFICATION:{YES|NO}]
    [NON_RECEIPT_NOTIFICATION:{YES|NO}]
END
```

(×) For FETCH results both, *common name* <u>and</u> *personal name* is possible.

Form 3:

```
X400F3::
    COUNTRY:<country>
    ADMD:<admd>
    [PRMD:<prmd>]
    X121_ADDRESS:<x121_string>
    [FREEFORMNAME:<freeform_name>]
    [TELEPHONENUMBER:<telephone_number>]
    [DDALIST::
        DDA::DDA_TYPE:<ddatype> DDA_VALUE:<ddavalue> END
        [DDA::DDA_TYPE:<ddatype> DDA_VALUE:<ddavalue> END]
        [DDA::DDA_TYPE:<ddatype> DDA_VALUE:<ddavalue> END]
        [DDA::DDA_TYPE:<ddatype> DDA_VALUE:<ddavalue> END]
    END]
    [RECEIPT_NOTIFICATION:{YES|NO}]
    [NON_RECEIPT_NOTIFICATION:{YES|NO}]
END
```

The address specifications are as follows:

COUNTRY:<country>   *country* specifies one of a predefined set of country codes identifying the country (e.g. DE for Germany).

ADMD:<admd>   *admd* specifies an administrative management domain (PTT authority) within the country.

UA_CONT_ID:<uaid>   *uaid* specifies the intended recipient in the form of a unique User Agent identification number. This specification only occurs with form 2 address specifications.

X121_ADDRESS:<x121_string>
                         x121_string specifies the intended recipient in the form of an X.121-style terminal address. This specification only occurs with form 3 address specifications.

PRMD:<prmd>   *prmd* specifies a private management domain (non-PTT organization) within the country and subject to the ADMD.

SURNAME:<surname>   *surname* specifies the surname of the recipient. This specification only occurs with form 1 address specifications.

GIVENNAME:<givenname>
                         givenname specifies the given name (first name) of the intended recipient. This specification only occurs with form 1 address specifications.

INITIALS<initials>   *initials* specifies the initials of the intended recipient. This specification only occurs with form 1 address specifications.

GENERATION:<generation>
                         generation specifies a generation title (Jr. Sr. etc.) for the intended recipient. This specification only occurs with form 1 address specifications.

COMMON_NAME:<common_name>
                         common_name specifies the common name of the intended recipient. This specification only occurs with form 1 address specifications.

ORGANIZATION:<orgname>

> *orgname* specifies the name of the organization of which the recipient is a member. This specification only occurs with form 1 address specifications.

ORG_UNIT:<org_unit>

> *org_unit* specifies the name of a subdivision of the organization specified under *ORGANIZATION*. Up to 4 organizational units can be specified. This specification only occurs with form 1 address specifications.

FREEFORMNAME:<freeform_name>

> *freeform_name* specifies a free-form name, which is not used for routing.

TELEPHONENUMBER:<telephone_number>

> *telephone_number* specifies the telephone number of the recipient.

DDA_TYPE:<ddatype>

DDA_VALUE:<ddavalue> *ddatype* and *ddavalue* together specify a domain-defined attribute, typically identifying a particular service available within the selected management domain. A typical example might be DDA_TYPE:"service" DDA_VALUE:"fax". This specification can occur up to four times to provide more detail.

RECEIPT_NOTIFICATION:{YES|NO}

> This parameter is used to specify whether a receipt notification was required or not.

NON_RECEIPT_NOTIFICATION:{YES|NO}

> This parameter is used to specify whether a Non-Receipt notification was required or not.

Example 1

This example demonstrates a simple *FETCH* command that fetches all the messages currently in *INFOLDER*. Since the result file is very extensive, its contents have been abbreviated in the example.

Contents of command script

```
$ FETCH INFOBASE: INFOLDER
```

Contents of result file

```
COMMAND: FETCH INFOBASE:INFOLDER
RESULT::
OBJECT::
FOLDER: "INFOLDER"
SEQ_NR: 00000000000000E8
ENTRY_STATUS: LISTED
TIME: 19950426044200Z
ENTRY_TYPE: IPM
MESSAGE_SIZE: 380
HAS_ATTACHMENTS: NO
ENVELOPE::
  MPDUID::
    COUNTRY: DE
    ADMD: OMSADMD
    ID: 4711ABCDEF0123456789A
  END
END
HEADER::
  IMPDU::
    IMPDU_HEADING::
      ENTRY_STATUS: LISTED
      TIME: 19950426044200Z
      UA_CONT_ID: "U01035-54QD3D7A6"
      PRIORITY: NORMAL
      IPM_ID::
        LOCAL_IPM_ID: "U01035-54QD3D7A6"
      END
      ORIGINATOR::
        X400F1::
          COUNTRY: "DE"
          ADMD: "OMSADMD"
          PRMD: "OMSPRMD"
          COMMON_NAME: "GRAHAM HILL"
          ORGANIZATION: "LONDON CAR COMPANY"
        END
      END
      SUBJECT: "Dates of in-house presentation"
      IMPORTANCE: 00000001
      PRIMARY_RECIPIENTS::
        X400F1::
          COUNTRY: "DE"
          ADMD: "OMSADMD"
          COMMON_NAME: "JACK STEWART"
        END
        X400F1::
          COUNTRY: "DE"
          ADMD: "OMSADMD"
```

```
                COMMON_NAME: "MANUEL FANGIO"
              END
            END
          END
        END
      END
      .
      .
      .
      OBJECT::
      FOLDER: "INFOLDER"
      SEQ_NR: 00000000000000ED
      ENTRY_STATUS: OLD
      TIME: 19950426044333Z
      ENTRY_TYPE: IPM
      MESSAGE_SIZE: 41608
      HAS_ATTACHMENTS: YES
      ENVELOPE::
        MPDUID::
          COUNTRY: DE
          ADMD: OMSADMD
          ID: 4711ABCDEF0123456789A
        END
      END
      HEADER::
        IMPDU::
          IMPDU_HEADING::
            ENTRY_STATUS: OLD
            TIME: 19950426044333Z
            UA_CONT_ID: "U01035-54QD3FB0B"
            PRIORITY: NORMAL
            IPM_ID::
              LOCAL_IPM_ID: "U01035-54QD3FB0B"
            END
            ORIGINATOR::
              X400F1::
                COUNTRY: "DE"
                ADMD: "OMSADMD"
                PRMD: "OMSPRMD"
                COMMON_NAME: "GRAHAM HILL"
                ORGANIZATION: "LONDON CAR COMPANY"
              END
            END
            SUBJECT: "Agenda for presentation day"
            IMPORTANCE: 00000001
            PRIMARY_RECIPIENTS::
              X400F1::
                COUNTRY: "DE"
                ADMD: "OMSADMD"
                PRMD: "OMSPRMD"
                PERSONAL_NAME::
                  SURNAME: "STEWART"
                END
                RECEIPT_NOTIFICATION: YES
                NON_RECEIPT_NOTIFICATION: YES
              END
              X400F2::
                COUNTRY: "DE"
                ADMD: "OMSADMD"
                UA_CONT_ID: "92002"
```

```
                END
              END
            END
          IMPDU_BODY::
            ATTACH::
              TYPE:"BIN"
              FILE:"BUA_BIN_ATT_DIR:fetch_example.001"
            END
          END
        END
      END
.
.
.
NUMBER_FETCHED_OBJECTS: 7
END
```

Example 2

This example demonstrates the use of two *FETCH* commands with selectors in order to fetch all the reports and receipt notifications currently in *INFOLDER*. Since the result file is very extensive, its contents have been abbreviated in the example.

Contents of command script

```
$ FETCH INFOBASE: INFOLDER ENTRY_TYPE: RPT
$ FETCH INFOBASE: INFOLDER ENTRY_TYPE: RN
```

Contents of result file

```
COMMAND: FETCH INFOBASE:INFOLDER ENTRY_TYPE: RPT
RESULT::
OBJECT::
FOLDER: "INFOLDER"
SEQ_NR: 00000000000000E2
ENTRY_STATUS: NEW
TIME: 19950426044219Z
ENTRY_TYPE: RPT
MESSAGE_SIZE: 0
HAS_ATTACHMENTS: NO
DR::
  DRPDU::
    X400F1::
      COUNTRY: "DE"
      ADMD: "OMSADMD"
      PRMD: "OMSPRMD"
      COMMON_NAME: "GRAHAM HILL"
      ORGANIZATION: "LONDON CAR COMPANY"
    END
    TIME: 19950426044219Z
    MPDUID::
      COUNTRY: DE
      ADMD: OMSADMD
      ID: 4711ABCDEF0123456789A
    END
    UA_CONT_ID: "U01035-54QD3DCD8"
    RECIPIENT_REPORT_INFO::
      REPORT_INFO::
        X400F1::
          COUNTRY: "DE"
```

```
                      ADMD: "OMSADMD"
                      PRMD: "OMSPRMD"
                      PERSONAL_NAME::
                         SURNAME: "STEWART"
                      END
                   END
                   ARRIVAL_DATE:: 19950426044219Z
                   DELIVER_DATE: 19950426044213Z
                END
                REPORT_INFO::
                   X400F1::
                      COUNTRY: "DE"
                      ADMD: "OMSADMD"
                      PRMD: "OMSPRMD"
                      PERSONAL_NAME::
                         SURNAME: "FANGIO"
                      END
                   END
                   ARRIVAL_DATE:: 35470418034646Z
                   DELIVER_DATE: 19950426044213Z
                END
                REPORT_INFO::
                   X400F1::
                      COUNTRY: "DE"
                      ADMD: "OMSADMD"
                      PRMD: "OMSPRMD"
                      PERSONAL_NAME::
                         SURNAME: "WINKELHOCK"
                      END
                   END
                   ARRIVAL_DATE:: 35470418034646Z
                   DELIVER_DATE: 19950426044213Z
                END
                REPORT_INFO::
                   X400F1::
                      COUNTRY: "DE"
                      ADMD: "OMSADMD"
                      PRMD: "OMSPRMD"
                      PERSONAL_NAME::
                         SURNAME: "VILLENEUVE"
                      END
                      ORGANIZATION: "VITESSE"
                   END
                   ARRIVAL_DATE:: 35470418034646Z
                   DELIVER_DATE: 19950426044213Z
                END
             END
          END
       END
       .
       .
       .
       NUMBER_FETCHED_OBJECTS: 4
       END
       COMMAND: FETCH INFOBASE:INFOLDER ENTRY_TYPE: RN
       RESULT::
       OBJECT::
       FOLDER: "INFOLDER"
       SEQ_NR: 00000000000000EC
       ENTRY_STATUS: NEW
```

```
      TIME: 19950426045050Z
      ENTRY_TYPE: RN
      MESSAGE_SIZE: 116
      HAS_ATTACHMENTS: NO
      ENVELOPE::
        MPDUID::
          COUNTRY: DE
          ADMD: OMSADMD
          ID: 4711ABCDEF0123456789A
        END
      END
      SR::
        RECEIPT_NOTIFICATION:
          X400F1::
            COUNTRY: "DE"
            ADMD: "OMSADMD"
            PRMD: "OMSPRMD"
            PERSONAL_NAME::
              SURNAME: "HILL"
            END
          END
          IPM_ID::
            LOCAL_IPM_ID: "U01035-54QD3FB0B"
          END
          RECEIPT: 19950426045048Z
          TYPE_OF_RECEIPT: AUTOMATIC
        END
      END
      END
      .
      .
      .
      NUMBER_FETCHED_OBJECTS: 3
      END
```

FILE

This command moves messages from one folder to another.

Box type: MAIL box.

Syntax

```
$FILE INFOBASE:<source_infobase> [selector]
      DEST_FOLDER:<target_infobase>
```

Parameters

INFOBASE:<source_infobase>

source_infobase can contain one of the following values:
SENTFOLDER, *INFOLDER*, *<user_defined_folder>*.
Refer to section 3.4 on page 16 for details of these specifications.

selector   The selector identifies the messages, which are to be moved. The available selectors and the message selector syntax are described on page 78.

Default: All messages in the selected infobase.

DEST_FOLDER:<target_infobase>

target_infobase must be *SENTFOLDER*, *INFOLDER* or a user-defined folder.

Description

The *FILE* command moves one or more messages from a system-defined folder or user-defined folder to a user-defined folder. The source folder is specified by *source_infobase*, and may be followed by a selector to indicate which messages are to be moved from the folder. The target folder is specified by *target_infobase*.

Result syntax

```
RESULT::
  NUMBER_OF_FILED_MESSAGES:<integer>
END
```

Result data

NUMBER_OF_FILED_MESSAGES:<integer>
                    integer returns the number of messages that have been moved.

Example

The *FILE* command in this example moves all messages in *SENTFOLDER* which contain the string "Mirrors" as part of their subject to the user folder *FN01*.

Contents of command script

```
$ FILE  INFOBASE: SENTFOLDER   SUBJECT CONTAINS "Mirrors"
DEST_FOLDER: FN01
```

Contents of result file

```
COMMAND: FILE INFOBASE:SENTFOLDER
    SUBJECT CONTAINS "Mirrors"      DEST_FOLDER: FN01
RESULT::
NUMBER_OF_FILED_MESSAGES: 5
END
```

LIST FOLDER

This command returns information on the names of folders and the number of objects they contain.

Box type: EDI box and MAIL box.

Syntax

$LIST INFOBASE:FOLDER

Parameters

The *LIST FOLDER* command has no parameters.

Description

The *LIST FOLDER* command returns information on the names of folders and the number of objects they contain. No selectors are permitted in the command.

Result syntax

```
RESULT::
  OBJECT::
    FOLDER:<folder_name>
    NUMBER_OBJECTS:<integer>
  END ...
END
```

Result data

FOLDER:<folder_name>   *folder_name* indicates the name of the listed folder.

NUMBER_OBJECTS:<integer>

                integer returns the number of objects contained in *folder_name*.

Example

The contents of the result file shown in this example are incomplete. The complete output includes data on the user folders *FN03* through *FN15*.

Contents of command script

```
$ LIST INFOBASE: FOLDER
```

Contents of result file

```
COMMAND: LIST INFOBASE:FOLDER
RESULT::
OBJECT::
FOLDER: "INFOLDER"
NUMBER_OBJECTS: 6
END
OBJECT::
FOLDER: "SENTFOLDER"
NUMBER_OBJECTS: 6
END
OBJECT::
FOLDER: "FN01"
NUMBER_OBJECTS: 0
END
OBJECT::
FOLDER: "FN02"
NUMBER_OBJECTS: 0
END
.
.
.
OBJECT::
FOLDER: "FN16"
NUMBER_OBJECTS: 0
END
END
```

LIST <message_folder>

This command returns information on the contents of a specified folder.

Box type: EDI box and MAIL box.

Syntax

```
$LIST INFOBASE:<infobase_name> [selector]
```

Parameters

INFOBASE:<infobase_name>

> *infobase_name* can contain one of the following values:
> *SENTFOLDER*, *INFOLDER*, *<user_defined_folder>* (MAIL box
> only), *MAILBOX* (EDI box only). Refer to section 3.4 on page 16
> for details of these specifications.

selector
> The selector identifies the entries, which are to be listed. The
> available selectors and the selector syntax are described on page
> 78.
>
> Default: All entries in the selected infobase.

Description

The *LIST <message_folder>* command is used to return information on messages in the
specified infobase.

Result syntax

```
RESULT::
  FOLDER:<folder_name>
  OBJECT::
    SEQ_NR:<sequence_no>
    ENTRY_STATUS:{NEW|LISTED|OLD|SENT}
    TIME:<time>
    ENTRY_TYPE:<entry_type>
    MESSAGE_SIZE:<integer>
    FIRST_RECIP_OR_ORIG:<identifier>
    SUBJECT:<subject_string>
    IPM_ID::LOCAL_IPM_ID:<ipm_id> END
    PRIORITY:{URGENT|NORMAL|NONURGENT}
    HAS_ATTACHMENTS:{YES|NO}
  END ...
  NUMBER_LISTED_OBJECTS:<integer>
END
```

Result data

FOLDER:<folder_name>   *folder_name* indicates the name of the folder in which the
                          messages are stored.

SEQ_NR:<sequence_no>   *sequence_no* returns the 16-digit hexadecimal sequence number assigned to the message.

ENTRY_STATUS:{NEW|LISTED|OLD|SENT}
                              indicates the entry status of the message.

TIME:<time>   For messages in *SENTFOLDER*, *time* indicates the submission time of the message and for incoming messages, *time* indicates the delivery time of the message. The supported date and time format is described in section 3.2 on page 14.

ENTRY_TYPE:<entry_type>
                              entry_type indicates the message type. Refer to appendix G on page 124 for details of the possible values.

MESSAGE_SIZE:<integer>
                              integer returns the size of the message in bytes. This item is not output for a delivery report.

FIRST_RECIP_OR_ORIG:<identifier>
                              identifier indicates the first recipient (for outgoing messages) or originator (for incoming messages) of the message and contains the most informative element of the address found by the Batch User Agent (typically the common name or surname). This item is not output for a delivery report.

SUBJECT:<subject_string>
                              subject_string contains the subject string as specified in the *SUBMIT* command. This item is not output for a delivery report.

ipm_id   contains the interpersonal message ID of the original message. This item is not output for a delivery report.

PRIORITY:{URGENT|NORMAL|NONURGENT}
                              specifies the priority of the message. This item is not output for a delivery report.

HAS_ATTACHMENTS:{YES|NO}
                              indicates whether the message has any attachments or not. This item is not output for a delivery report.

                              Note: If the first body part is the only body part and if this body part is a text body part, the value of *HAS_ATTACHMENTS* is *NO*.

Example

Contents of command script
```
$ LIST INFOBASE: INFOLDER
```

Contents of result file
```
COMMAND: LIST INFOBASE:INFOLDER
RESULT::
FOLDER: "INFOLDER"
OBJECT::
SEQ_NR: 00000000000000E8
ENTRY_STATUS: LISTED
TIME: 19950426044200Z
ENTRY_TYPE: IPM
MESSAGE_SIZE: 380
FIRST_RECIP_OR_ORIG: "GRAHAM HILL"
SUBJECT: "Quotation for body panels"
IPM_ID::
LOCAL_IPM_ID: "U01035-54QD3D7A6"
END
PRIORITY: NORMAL
HAS_ATTACHMENTS: NO
END
OBJECT::
SEQ_NR: 00000000000000E9
ENTRY_STATUS: LISTED
TIME: 19950426044214Z
ENTRY_TYPE: IPM
MESSAGE_SIZE: 969
FIRST_RECIP_OR_ORIG: "GRAHAM HILL"
SUBJECT: "Order confirmation"
IPM_ID::
LOCAL_IPM_ID: "U01035-54QD3DCD8"
END
PRIORITY: URGENT
HAS_ATTACHMENTS: NO
END
OBJECT::
SEQ_NR: 00000000000000ED
ENTRY_STATUS: LISTED
TIME: 19950426044333Z
ENTRY_TYPE: IPM
MESSAGE_SIZE: 41608
FIRST_RECIP_OR_ORIG: "GRAHAM HILL"
SUBJECT: "Cost sheets"
IPM_ID::
LOCAL_IPM_ID: "U01035-54QD3FB0B"
END
PRIORITY: NORMAL
HAS_ATTACHMENTS: YES
END
NUMBER_LISTED_OBJECTS: 3
END
```

LIST PROFILE

This command returns information on the current user profile.

Box type: EDI box.

Syntax

```
$LIST INFOBASE:PROFILE
```

Parameters

The *LIST PROFILE* command has no parameters.

Description

The *LIST PROFILE* command returns the number of days that EDI documents are to be archived as specified in the user profile. This is the only setting in the user profile that can be changed using the Batch User Agent.

Refer to Appendix B on page 105 for details on other profile settings which affect operation of the Batch User Agent.

Result syntax

```
RESULT::
  ARCHIVE_DAYS:<integer 0..365>]
END
```

Result data

ARCHIVE_DAYS:<integer 0..365>

> *integer* returns the current setting for the number of days that EDI documents are archived. Values between 0 and 365 are possible. A value of 0 indicates that EDI documents are archived for an indefinite period.

Example

Contents of command script
```
$ LIST INFOBASE: PROFILE
```

Contents of result file
```
COMMAND: LIST INFOBASE:PROFILE
RESULT::
ARCHIVE_DAYS: 200
END
```

LIST SUBSCRIBER

This command returns information on subscribers.

Box type: MAIL box.

Syntax

```
$LIST INFOBASE:SUBSCRIBER [filter [AND filter]]
```

Parameters

filter   *filter* identifies the addresses, which are to be listed. There are two
                            possible options for *filter*:

                            COMMON_NAME{: | CONTAINS} <string>
                                        string is the common name of the required
                                        subscriber. If you specify the operator *CONTAINS*,
                                        the common name must contain the string specified.
                                        Matching is case-insensitive for both operators.

                            ORGANIZATION:<org_name>
                                        org_name is the name of the organization where the
                                        required subscriber can be reached.

Description

The *LIST SUBSCRIBER* command returns address information for subscribers. This command
allows BUA users to access the X.500 directory services. The number of addresses returned is
restricted by the address limit specified in the user profile.

Result syntax

```
RESULT::
  OBJECT::
    x400_address
  END ...
END
```

Result data

x400_address   *x400_address* specifies the address of the subscriber. Refer to the
                            FETCH command on page 54 for details of the syntax of
                            x400_address.

Example

Contents of command script
```
$ LIST INFOBASE: SUBSCRIBER COMMON_NAME: "JACK STEWART"
```

Contents of result file
```
COMMAND: LIST INFOBASE:SUBSCRIBER COMMON_NAME: "JACK STEWART"
RESULT::
OBJECT::
  X400F1::
    COUNTRY: "DE"
    ADMD: "OMSADMD"
    PRMD: "OMSPRMD"
    PERSONAL_NAME::
      SURNAME: "STEWART"
      GIVENNAME: "JACK"
      INITIALS: "C"
    END
    ORGANIZATION: "FASTCARS"
    ORG_UNIT_HIERARCHY::
      ORG_UNIT: "SALES"
    END
  END
END
NUMBER_LISTED_SUBSCRIBERS: 1
END
```

MODIFY

This command is used to change the name of a user folder.

Box type: MAIL box.

Syntax

```
$MODIFY INFOBASE:<infobase_name> OLDNAME:<old_name>
        NEWNAME:<new_name>
```

Parameters

INFOBASE:<infobase_name>

> *infobase_name* can contain only the value *FOLDER*. Refer to section 3.4 on page 16 for details of this specification.

OLDNAME:<old_name> *old_name* specifies the name of the user-defined folder whose name is to be changed.

NEWNAME:<new_name> *new_name* specifies the new name for the folder. The maximum length of *new_name* is 14 characters.

Description

The *MODIFY* command changes the name of a private folder (initial default names *FN01* through *FN16*).

You cannot use the *MODIFY* command to rename system folders. *old_name* must refer to an existing folder.

new_name cannot already be the name of an existing folder (including the names of the system folders: *INFOLDER*, *SENTFOLDER*, *FOLDER*, *DOMAIN*, *SUBSCRIBER*, *PROFILE*, *MAILBOX*, *EDIOUTFOLDER*, *EDIINFOLDER*), must be no more than 14 characters long and must include at least one letter or digit. Leading and trailing blanks are ignored.

Result syntax

```
RESULT::
  FOLDER_MODIFIED::
    FOLDER:<old_name>
    NEWNAME:<new_name>
  END
END
```

Result data

FOLDER:<old_name>

> *old_name* specifies the old name of the user-defined folder.

NEWNAME:<new_name>

> *new_name* specifies the new name of the user-defined folder.

Example

The MODIFY command shown here changes the default user folder name *FN01* to *ORDERS*.

Contents of command script

```
$ MODIFY INFOBASE: FOLDER OLDNAME: FN01 NEWNAME: ORDERS
```

Contents of result file

```
COMMAND: MODIFY INFOBASE:FOLDER OLDNAME: FN01 NEWNAME: ORDERS
RESULT::
FOLDER_MODIFIED::
FOLDER: "FN01"
NEWNAME: "ORDERS"
END
END
```

REGISTER

This command is used to change the number of days for which EDI documents are archived.

Box type: EDI box.

Syntax

$REGISTER [INFOBASE:<infobase_name>] ARCHIVE_DAYS:<integer>

Parameters

INFOBASE:<infobase_name>

> *infobase_name* can contain only the value *PROFILE*. Refer to section 3.4 on page 16 for details of this specification.

ARCHIVE_DAYS:<integer>

> *integer* specifies the number of days for which EDI documents are archived. The value specified may be between 0 and 365 inclusive. If you specify 0, the archive period will be unlimited.

Description

The *REGISTER* command is used to change the value for the number of days EDI documents are archived in the local profile. This is the only item in the local profile that can be changed by the subscriber. EDI documents with the status *SENT* or *OLD* will be deleted after the specified number of days has expired.

Refer to Appendix B on page 105 for details on other profile settings which affect operation of the Batch User Agent.

Result syntax

```
RESULT::
  ARCHIVE_DAYS:<integer 0..365>]
END
```

Result data

ARCHIVE_DAYS:<integer 0..365>

> *integer* returns the new setting for the number of days that EDI documents are archived. Values between 0 and 365 are possible. A value of 0 indicates that EDI documents are archived for an indefinite period.

Example

Contents of command script
```
$ REGISTER   ARCHIVE_DAYS: 200
```

Contents of result file
```
COMMAND: REGISTER ARCHIVE_DAYS: 200
RESULT::
ARCHIVE_DAYS: 200
END
```

REMARK

This command is used to insert comments in a command script.

Box type: EDI box and MAIL box.

Syntax

$REMARK <remark>

Parameters

remark   *remark* is a string containing any printable characters. Unlike other
                                strings, a remark string is terminated by an end-of-line character.

Description

The *REMARK* command is used to add comment lines to a command script. Each new
comment line must be preceded by the *REMARK* command. Comments extend to the end of the
current line. The comment itself is ignored and no result data is produced, although the
command is written to the result file.

Restrictions

The length of a comment must not exceed 1 line.

Example

Contents of command script
```
$ REMARK This is a comment
```

Contents of result file
```
COMMAND: REMARK This is a comment
```

selector

selector identifies the messages, which are to be processed. It is not a command as such, but it is a parameter common to the commands which access messages in the infobases and is therefore described once here, rather than being repeated in each relevant command description.

Box type: EDI box and MAIL box.

Syntax

```
{SEQ_NR:<sequence_number>|CURRENT|ALL|
[range] [filter [AND filter] ...]}
```

Parameters

SEQ_NR:<sequence_number>

sequence_number is a 16-digit hexadecimal number that selects the message with the specified sequence number from the specified infobase.

CURRENT

selects the current message. *CURRENT* is defined by using the *FETCH* or *DOWNLOAD MESSAGE* command. If you used these commands with the selector *ALL*, *CURRRENT* is set to the last processed message. If you have not yet fetched or downloaded a message *CURRENT* is not a valid selector.

ALL

selects all messages in the specified infobase.

range

selects a range of messages from the specified infobase. Two alternative range specifications are available: *NUMBER_RANGE* and *TIME_RANGE*.

NUMBER_RANGE::FROM:<int1> TO:<int2> END

This specification selects all the messages with sequence numbers in the range from *int1* to *int2* inclusive. *int1* and *int2* are integers specified in hexadecimal form.

TIME_RANGE::AFTER:<time1> BEFORE:<time2> END

This specification selects all the messages with submission dates in the range from *time1* to *time2* inclusive. The supported date and time format is described in section 3.2 on page 14.

filter

selects messages within the specified range whose attributes match those in the filter. The permitted specifications are listed below. When a number of different filter specifications are made, the specifications must be explicitly ANDed. No duplicates are permitted.

ENTRY_STATUS:<entry_status>
          selects messages with the given status. The possible
          values for *entry_status* are: *NEW*, *LISTED*, *OLD*
          and *SENT*.

LOCAL_IPM_ID {:|CONTAINS} <string>
          selects messages with the interpersonal message
          identifier specified by string. If you specify the
          colon operator (:), the interpersonal message
          identifier must exactly match the string specified. If
          you specify the *CONTAINS* operator, the
          interpersonal message identifier must contain the
          string specified. The *CONTAINS* operator is not
          case-sensitive.

ENTRY_TYPE:<entry_type>
          selects messages of the specified message type. The
          supported types are: *IPM, EDIM, RPT, RN,* and
          NRN. Refer to appendix G on page A-135 for
          details of the possible values.

FIRST_RECIP_OR_ORIG {:|CONTAINS} <string>
          selects sent messages where the specification in
          string denotes the first recipient, or received
          messages where *string* denotes the originator of the
          message. If you specify the colon operator (:), the
          first recipient or originator must exactly match the
          string specified. If you specify the *CONTAINS*
          operator, the first recipient or originator must
          contain the string specified. Matching is case-
          insensitive for both operators.

PRIORITY:<priority>
          selects messages with the given priority. The
          possible values for *priority* are: *NONURGENT*,
          NORMAL and *URGENT.*

SUBJECT {:|CONTAINS} <string>
          selects messages where the subject of the message
          corresponds to the specification in *string*. If you
          specify the colon operator (:), the subject must
          exactly match the string specified. If you specify the
          CONTAINS operator, the subject must contain the
          string specified. The *CONTAINS* operator is not
          case-sensitive.

TIME:<time>
          selects messages with the submission date specified.
          The supported date and time format is described in
          section 3.2 on page 14.

Description

Note on defaults

The default value for *selector* varies from command to command and is indicated at the
appropriate point in each relevant command description.

Example 1

This first example shows the *LIST* command used with the *ALL* selector. Since the result file for this command is extensive, it has been abbreviated for the purposes of the example.

Contents of command script

```
$ LIST INFOBASE:INFOLDER ALL
```

Contents of result file

```
RESULT::
FOLDER: "INFOLDER"
OBJECT::
SEQ_NR: 00000000000000E8
ENTRY_STATUS: LISTED
TIME: 19950426044200Z
ENTRY_TYPE: IPM
MESSAGE_SIZE: 380
FIRST_RECIP_OR_ORIG: "JACK STEWART"
SUBJECT: "Order correction"
IPM_ID::
LOCAL_IPM_ID: "U01035-54QD3D7A6"
END
PRIORITY: NORMAL
HAS_ATTACHMENTS: NO
END
OBJECT::
SEQ_NR: 00000000000000E9
ENTRY_STATUS: LISTED
TIME: 19950426044214Z
ENTRY_TYPE: IPM
MESSAGE_SIZE: 969
FIRST_RECIP_OR_ORIG: "JACK STEWART"
SUBJECT: "New delivery address"
IPM_ID::
LOCAL_IPM_ID: "U01035-54QD3DCD8"
END
PRIORITY: URGENT
HAS_ATTACHMENTS: NO
END
.
.
.
OBJECT::
SEQ_NR: 00000000000000F0
ENTRY_STATUS: LISTED
TIME: 19950426044446Z
ENTRY_TYPE: IPM
MESSAGE_SIZE: 73270
FIRST_RECIP_OR_ORIG: "JACK STEWART"
SUBJECT: "New parts catalog now available"
IPM_ID::
LOCAL_IPM_ID: "U01033-54QD416BC"
END
PRIORITY: NORMAL
HAS_ATTACHMENTS: NO
END
NUMBER_LISTED_OBJECTS: 9
END
```

Example 2

This example uses *LIST* command with the *SUBJECT*, *PRIORITY* and *ENTRY_STATUS*
selectors to restrict output to the required messages. In this case, output is restricted to a single
message.

Contents of command script

```
$ LIST INFOBASE:INFOLDER SUBJECT CONTAINS "OK"
      AND PRIORITY: URGENT
      AND ENTRY_STATUS: LISTED
```

Contents of result file

```
COMMAND: LIST INFOBASE:INFOLDER
RESULT::
FOLDER: "INFOLDER"
OBJECT::
SEQ_NR: 00000000000000E9
ENTRY_STATUS: LISTED
TIME: 19950426044214Z
ENTRY_TYPE: IPM
MESSAGE_SIZE: 969
FIRST_RECIP_OR_ORIG: "JACK STEWART"
SUBJECT: "New delivery date OK"
IPM_ID::
LOCAL_IPM_ID: "U01035-54QD3DCD8"
END
PRIORITY: URGENT
HAS_ATTACHMENTS: NO
END
NUMBER_LISTED_OBJECTS: 1
END
```

STATUS

This command generates an activity report.

Box type: EDI box.

Syntax

```
$STATUS INFOBASE:<infobase_name> [selector]
        [SUMMARY|DETAIL|CONTENTS]
```

Parameters

INFOBASE:<infobase_name>

> *infobase_name* can contain one of the following values: *SENTFOLDER*, *INFOLDER*. Refer to section 3.4 on page 16 for details of these specifications.

selector
> The selector identifies the entries, which are to be listed. The available selectors and the selector syntax are described on page 78.
>
> Default: All entries in the selected infobase.

SUMMARY
> (default). Specifies that a summary EDI activity report is to be created.

DETAIL
> specifies that a detailed EDI activity report is to be generated.

CONTENTS
> specifies that an EDI file cabinet contents report is to be generated.

Description

The *STATUS* command is used to generate an activity report. This report takes the form of a mail message and is stored in the *MAILBOX* folder. The report can then be read in the same way as any other message.

Result syntax

```
RESULT::
  STATUS_TYPE:{SUMMARY|DETAIL|CONTENTS}
  SEQ_NR:<sequence_no>
END
```

Result data

STATUS_TYPE:{SUMMARY|DETAIL|CONTENTS}

> indicates the type of report which has been created.

> SUMMARY   specifies that a summary EDI activity report has been generated.

> DETAIL   specifies that a detailed EDI activity report has been generated.

> CONTENTS   specifies that an EDI file cabinet contents report has been generated.

SEQ_NR:<sequence_no>   *sequence_no* returns the 16-digit hexadecimal sequence number of the message in which the report has been stored. This number will always be 0000000000000000.

Example

Contents of command script

```
$ STATUS INFOBASE: SENTFOLDER SUMMARY
$ STATUS INFOBASE: SENTFOLDER CONTENTS
$ STATUS INFOBASE: INFOLDER DETAIL
```

Contents of result file

```
COMMAND:  STATUS INFOBASE:SENTFOLDER SUMMARY
RESULT::
STATUS_TYPE: SUMMARY
SEQ_NR: 0000000000000000
END
COMMAND:  STATUS INFOBASE:SENTFOLDER CONTENTS
RESULT::
STATUS_TYPE: CONTENTS
SEQ_NR: 0000000000000000
END
COMMAND:  STATUS INFOBASE:INFOLDER DETAIL
RESULT::
STATUS_TYPE: DETAIL
SEQ_NR: 0000000000000000
END
```

The contents of the three messages is shown below:

```
COMMAND:   STATUS INFOBASE:SENTFOLDER SUMMARY

no.   status   creation           typ    name           subject/id       Kb
    1  SENT     11-APR-1996:09:07   EDIM   T: EMS1500133 EDIFACT EDI_1_2    1
    2  SENT     11-APR-1996:09:07   EDIM   T: EMS1500133 EDIFACT EDI_1_2    1
    3  SENT     11-APR-1996:09:07   EDIM   T: EMS1500133 EDIFACT EDI_1_2    1



COMMAND:   STATUS INFOBASE:SENTFOLDER CONTENTS

no.   status   creation           typ    name           subject/id       Kb
    1  SENT     11-APR-1996:09:07   EDIM   T: EMS1500133 EDIFACT EDI_1_2    1
    2  SENT     11-APR-1996:09:07   EDIM   T: EMS1500133 EDIFACT EDI_1_2    1
    3  SENT     11-APR-1996:09:07   EDIM   T: EMS1500133 EDIFACT EDI_1_2    1


COMMAND:   STATUS INFOBASE:SENTFOLDER DETAIL

nr     creation           name                      K
1      11-APR-1996:09:07   EMS1500133                1
        Sender ID:          EDI1500131
        Recipient ID:       EDI1500132
        ControlReference:   EDI_1_2
        IPM-ID:             E00002718-000000
        Subject:            EDIFACT EDI_1_2
2      11-APR-1996:09:07   EMS1500133                1
        Sender ID:          EDI1500131
        Recipient ID:       EDI1500132
        ControlReference:   EDI_1_2
        IPM-ID:             E00002719-000000
        Subject:            EDIFACT EDI_1_2
3      11-APR-1996:09:07   EMS1500133                1
        Sender ID:          EDI1500131
        Recipient ID:       EDI1500132
        ControlReference:   EDI_1_2
        IPM-ID:             E0000271a-000000
        Subject:            EDIFACT EDI_1_2
```

STORAGE

This command returns the number of storage units currently used.

Box type: EDI box and MAIL box.

Syntax

```
$STORAGE
```

Parameters

The *STORAGE* command has no parameters.

Description

You use the *STORAGE* command to find out how much disk space you are occupying on the OpenMS server. The storage space occupied is expressed in storage units. A storage unit is 512 bytes. You need this information in order to be able to cross-check for billing purposes.

STORAGE does not compute an up-to-the-minute value for disk occupancy. It simply reads the last value computed by the server, which is used for monthly storage. This is generally all you need to know, as you mostly just want to verify your monthly storage bill.

Result syntax

```
RESULT::
  NUMBER_USED_STORAGE_UNITS:<integer>
  NUMBER_NEW_STORAGE_UNITS:<integer>
  NUMBER_LISTED_STORAGE_UNITS:<integer>
  NUMBER_OLD_STORAGE_UNITS:<integer>
  STORAGE_TIME:<storage_time>
END
```

Result data

NUMBER_USED_STORAGE_UNITS:<integer>

> *integer* returns the total number of storage units occupied by the user.

NUMBER_NEW_STORAGE_UNITS:<integer>

> *integer* returns the number of storage units occupied by messages with the status *NEW*.

NUMBER_LISTED_STORAGE_UNITS:<integer>

> *integer* returns the number of storage units occupied by messages with the status *LISTED*.

NUMBER_OLD_STORAGE_UNITS:<integer>

> *integer* returns the number of storage units occupied by messages with the status *OLD*.

STORAGE_TIME:<storage_time>

> *storage_time* returns the time at which the storage values were calculated.

Example

Contents of command script

```
$ STORAGE
```

Contents of result file

```
COMMAND: STORAGE
RESULT::
NUMBER_USED_STORAGE_UNITS: 1
NUMBER_NEW_STORAGE_UNITS: 1
NUMBER_LISTED_STORAGE_UNITS: 0
NUMBER_OLD_STORAGE_UNITS: 0
STORAGE_TIME: 19950426000016Z
END
```

SUBMIT

This command creates and sends a message.

Box type: MAIL box.

Syntax

```
$SUBMIT [INFOBASE:<infobase_name>]
        [CONTENT_TYPE:{IPM84|IPM88}] message
```

Parameters

INFOBASE:<infobase_name>

> *infobase_name* can contain only the value SENTFOLDER. Refer to section 3.4 on page 16 for details of this specification.

CONTENT_TYPE:{IPM84|IPM88}

> selects the interpersonal message content type complying with the 1984 or 1988 CCITT Message Handling System recommendations respectively (see **Description** below and section 1.1).

message

> *message* is a sequence of keywords and data that together define the message to be sent. The relevant structure and syntax are described below.

Description

The *SUBMIT* command is used to create a message and send it to one or more other subscribers. All the data required for creating and structuring the message is included directly as a parameter of the *SUBMIT* command in the command script.

The structure of the message data must conform to the rules described below.

Input files are deleted after they have been processed successfully.

The interpersonal messages that you can send with SUBMIT are of two types. One content type complies with the 1984 version of the CCITT message handling system recommendations, the other with the 1988 version. You can choose a type either explicitly by using the *CONTENT_TYPE* specification (see above) or implicitly by omitting the specification, in which case the default content type defined in your user profile applies automatically. The 1988 content type accepts extra address forms. The differences will be pointed out in the course of this description.

The character set used depends on the default settings in the user profile for the content type (*IPM84* or *IPM88*) and for the default bodypart (*ISO* or *IA5*). The ISO Latin 1 character set is only used if the user profile contains the specification *ISO* for the default bodypart and if the user profile contains the specification *IPM88* for the default content type, or if the specification *IPM84* has been overridden in the *CONTENT_TYPE* qualifier of the *SUBMIT* command. In all other cases, the International Alphabet Number 5 (IA5) is used.

Structure of a message

The following syntax diagrams show the structure of a message as it must appear in the command script. However, the elements in the IMPDU_HEADING structure can be specified in any order.

```
IMPDU::

    IMPDU_HEADING::
        [PRIORITY:{NONURGENT|NORMAL|URGENT}]
        [DEFER_DATE:<time>]
        ORIGINATOR::x400_address END
        [SUBJECT:<subject_string>]
        PRIMARY_RECIPIENTS::x400_address END ...
        | &
        COPY_RECIPIENTS::x400_address END ...
        [REPLY_TO_USERS::x400_address END] ...
        [IN_REPLY_TO::
            UA_CONT_ID:<ipm_id>
        END]
    END

    [IMPDU_BODY::
        [{TEXT:<message_text> EOTEXT|TEXTFILE:<filename>}]
        [{ATTACH::|BODYPART::}
            TYPE:{TXT|BIN}
            FILE:<filename>
        END] ...
    END]

END
```

The parameters are as follows:

PRIORITY:{NONURGENT|NORMAL|URGENT}

> Here you define the message priority. This specification is optional.

DEFER_DATE:<time>   *time* defines a date and time at which your message is to be submitted. This specification is optional. By default the message is sent immediately. The supported date/time format is described in section 3.2 on page 14.

ORIGINATOR::x400_address END

> This specification provides addressing information on the sender (originator) of the message. The structure of *x400_address* is described below. This specification is mandatory. Form 3 addresses are not permitted.

SUBJECT:<subject_string>

> Here you define a subject for your message. This specification is optional, and no default is defined. The subject can be any explanatory string up to 128 characters long.

PRIMARY_RECIPIENTS::x400_address END

> Here you define the name and address of the primary recipient or recipients. You must define at least one primary recipient and can define any number of further primary recipients. The structure of *x400_address* is described below.

COPY_RECIPIENTS::x400_address END

> Here you define the names and addresses of carbon-copy recipients. Any number of carbon-copy recipients may be defined,

but it is also possible for none to be defined. The structure of *x400_address* is described below.

REPLY_TO_USERS::x400_address END

> This specification contains the names and addresses of preferred recipients of any reply to this message. Any number of preferred reply recipients may be defined, but it is also possible for none to be defined. The structure of *x400_address* is described below.

IN_REPLY_TO::UA_CONT_ID:<ipm_id> END

> If you are replying to a message you have received, *ipm_id* is where you specify the interpersonal message ID of this message.

TEXT:<message_text> EOTEXT

> *message_text* is a string where you specify the text you are sending. The message text can extend over a number of lines and is terminated by the keyword *EOTEXT*.

TEXTFILE:<filename>

> The text of a message can be contained in a file instead of being located directly in the script. *filename* specifies the name of the file containing the text.

{ATTACH::|BODYPART::}TYPE:{TXT|BIN} FILE:<filename> END

> Here you can define an attachment for your message. An attachment in this context is a file (typically a graphics file, spreadsheet etc.) appended to the message. This specification is optional.
>
> If you define an attachment, you must first specify the type. Two values are possible here: *TXT* (text file) or *BIN* (binary file). *filename* specifies the file to be attached.

The syntax of *x400_address* can take three different forms. The three forms are similar in content and the specifications are described together following the three syntax diagrams. The elements within the X400F* structures can be specified in any order.

Form 1:

```
X400F1::
    COUNTRY:<country>
    ADMD:<admd>
    [PRMD:<prmd>]
    {[COMMON_NAME:<common_name>]   (x)
     |
     [PERSONAL_NAME::
        SURNAME:<surname>
        [GIVENNAME:<givenname>]
        [INITIALS:<initials>]
        [GENERATION:<generation>]
     END]}
    [ORGANIZATION:<orgname>]
    [ORG_UNIT_HIERARCHY::
        ORG_UNIT:<org_unit> [ORG_UNIT:<org_unit>]
        [ORG_UNIT:<org_unit>] [ORG_UNIT:<org_unit>]
    END]
    [FREEFORMNAME:<freeform_name>]
    [TELEPHONENUMBER:<telephone_number>]
    [DDALIST::
        DDA::DDA_TYPE:<ddatype> DDA_VALUE:<ddavalue> END
        [DDA::DDA_TYPE:<ddatype> DDA_VALUE:<ddavalue> END]
        [DDA::DDA_TYPE:<ddatype> DDA_VALUE:<ddavalue> END]
        [DDA::DDA_TYPE:<ddatype> DDA_VALUE:<ddavalue> END]
    END]
    [DELIVERY_NOTIFICATION:{YES|NO}]
    [RECEIPT_NOTIFICATION:{YES|NO}]
    [NON_RECEIPT_NOTIFICATION:{YES|NO}]
END
```

Form 2:

```
X400F2::
    COUNTRY:<country>
    ADMD:<admd>
    [PRMD:<prmd>]
    UNIQUE_UA_ID:<uaid>
    [FREEFORMNAME:<freeform_name>]
    [TELEPHONENUMBER:<telephone_number>]
    [DDALIST::
        DDA::DDA_TYPE:<ddatype> DDA_VALUE:<ddavalue> END
        [DDA::DDA_TYPE:<ddatype> DDA_VALUE:<ddavalue> END]
        [DDA::DDA_TYPE:<ddatype> DDA_VALUE:<ddavalue> END]
        [DDA::DDA_TYPE:<ddatype> DDA_VALUE:<ddavalue> END]
    END]
    [DELIVERY_NOTIFICATION:{YES|NO}]
    [RECEIPT_NOTIFICATION:{YES|NO}]
    [NON_RECEIPT_NOTIFICATION:{YES|NO}]
END
```

(x) For FETCH results both, *common name* and *personal name* is possible.

Form 3:

```
X400F3::
    COUNTRY:<country>
    ADMD:<admd>
    [PRMD:<prmd>]
    X121_ADDRESS:<x121_string>
    [FREEFORMNAME:<freeform_name>]
    [TELEPHONENUMBER:<telephone_number>]
    [DDALIST::
        DDA::DDA_TYPE:<ddatype> DDA_VALUE:<ddavalue> END
        [DDA::DDA_TYPE:<ddatype> DDA_VALUE:<ddavalue> END]
        [DDA::DDA_TYPE:<ddatype> DDA_VALUE:<ddavalue> END]
        [DDA::DDA_TYPE:<ddatype> DDA_VALUE:<ddavalue> END]
    END]
    [DELIVERY_NOTIFICATION:{YES|NO}]
    [RECEIPT_NOTIFICATION:{YES|NO}]
    [NON_RECEIPT_NOTIFICATION:{YES|NO}]
END
```

The address specifications are as follows:

COUNTRY:<country> *country* specifies one of a predefined set of country codes identifying the country (e.g. DE for Germany). This specification is mandatory.

ADMD:<admd> *admd* specifies an administrative management domain (PTT authority) within the country. This specification is mandatory.

UNIQUE_UA_ID:<uaid> *uaid* specifies the intended recipient in the form of a unique User Agent identification number. This specification is only valid for form 2 address specifications, where it is mandatory.

X121_ADDRESS:<x121_string>
  x121_string specifies the intended recipient in the form of an X.121-style terminal address. This specification is only valid for form 3 address specifications, where it is mandatory (see note on restrictions below).

PRMD:<prmd> *prmd* specifies a private management domain (non-PTT organization) within the country and subject to the ADMD. This specification is not valid for forms 2 and 3 with IPM84 messages.

SURNAME:<surname> *surname* specifies the surname of the intended recipient. This specification is only valid for form 1 address specifications.

GIVENNAME:<givenname>
  givenname specifies the given name (first name) of the intended recipient. This specification is only valid for form 1 address specifications, where it is optional.

INITIALS:<initials> *initials* specifies the initials of the intended recipient. This specification is only valid for form 1 address specifications, where it is optional.

GENERATION:<generation>

>*generation* specifies a generation title (Jr., Sr. etc.) for the intended recipient. This specification is only valid for form 1 address specifications, where it is optional.

COMMON_NAME:<common_name>

>*common_name* specifies the common name of the intended recipient. This specification is available for form 1 IPM88 messages only.

ORGANIZATION:<orgname>

>*orgname* specifies the name of the organization at which the intended recipient can be reached. This specification is only valid for form 1 address specifications, where it is optional.

ORG_UNIT:<org_unit>   *org_unit* specifies the name of a subdivision of the organization specified under *ORGANIZATION*. Up to 4 organizational units can be specified. This specification is only valid for form 1 address specifications, where it is optional.

FREEFORMNAME:<freeform_name>

>*freeform_name* specifies a free-form name, which is not used for routing. This specification is optional.

TELEPHONENUMBER:<telephone_number>

>*telephone_number* specifies the telephone number of the intended recipient. This specification is optional.

DDA_TYPE:<ddatype>

DDA_VALUE:<ddavalue> *ddatype* and *ddavalue* together specify a domain-defined attribute, typically identifying a particular service available within the selected management domain. A typical example might be DDA_TYPE:"service" DDA_VALUE:"fax". This specification can be made up to four times to enable you to give more detail.

DELIVERY_NOTIFICATION:{YES|<u>NO</u>}

>This parameter is used to specify whether you require a delivery notification or not. The parameter is optional and is not available when specifying the address of the originator.

RECEIPT_NOTIFICATION:{YES|<u>NO</u>}

>This parameter is used to specify whether you require a receipt notification or not. The parameter is optional and is not available when specifying the address of the originator.

NON_RECEIPT_NOTIFICATION:{YES|<u>NO</u>}

>This parameter is used to specify whether you require a Non-Receipt notification or not. The parameter is optional and is not available when specifying the address of the originator.

Restrictions

The general OpenMS restrictions apply.

Addresses in form 3 are not permitted for the originator.

For security reasons, the data for the originator's address is always required as input. This data will, however always correspond to that of the subscriber as logged in via the BUA command script.

Result syntax

```
RESULT::
  IPM_ID::
    LOCAL_IPM_ID:<ipm_id>
  END
  MPDU_ID:<mpdu_id>
  SEQ_NR:<sequence_no>
  SUBJECT:<subject_string>
  FIRST_RECIP_OR_ORIG:<identifier>
  [SUBMIT_TIME:<time>]
  [ATTACH::TYPE:{TXT|BIN} FILE:<filename> END ...]
END
```

Result data

LOCAL_IPM_ID:<ipm_id>

ipm_id returns the interpersonal message ID of the submitted message.

MPDU_ID:<mpdu_id>   message ID, as returned by the Mail Transfer Agent (MTA).

SEQ_NR:<sequence_no>   *sequence_no* returns the sequence number of the submitted message (as a 16-digit hexadecimal number).

SUBJECT:<subject_string>

subject_string contains the subject string as specified in the *SUBMIT* command.

FIRST_RECIP_OR_ORIG:<identifier>

identifier indicates the first recipient of the message and contains the most informative element of the address found by the Batch User Agent (typically the common name or surname).

SUBMIT_TIME:<time>

time indicates the submission time of the message. The supported date and time format is described in section 3.2 on page 14.

TYPE:{TXT|BIN}   specifies whether the attachment is a text or binary file.

FILE:<filename>   *filename* specifies the name of the file containing the attachment.

Example 1

This example demonstrates the use of the *SUBMIT* command to send a message including a text attachment. The example uses address form 1 for the primary recipient and address form 2 for the copy recipient.

Contents of command script

```
$SUBMIT INFOBASE: SENTFOLDER
IMPDU::
  IMPDU_HEADING::
    PRIORITY: NORMAL
    ORIGINATOR::
      X400F1::
        COUNTRY:        DE
        ADMD:           OMSADMD
        PERSONAL_NAME::
          SURNAME:    "HILL"
        END
      END
    END
    PRIMARY_RECIPIENTS::
      X400F1::
        COUNTRY:        DE
        ADMD:           OMSADMD
        PRMD:           OMSPRMD
        PERSONAL_NAME::
          SURNAME:    "STEWART"
        END
      END
    END
    COPY_RECIPIENTS::
      X400F2::
        COUNTRY:        DE
        ADMD:           OMSADMD
        UNIQUE_UA_ID: 92002
      END
    END
    SUBJECT: "Invitation to visit our new premises"
  END
  IMPDU_BODY::
    ATTACH::
      TYPE: TXT
      FILE: invite.txt
    END
  END
END
```

Contents of result file

```
COMMAND: SUBMIT INFOBASE:SENTFOLDER
RESULT::
IPM_ID::
LOCAL_IPM_ID: "U01033-54QD416BC"
END
MPDU_ID: "126FF83611CE7F8C00AA1189"
SEQ_NR: 0000000000000018
SUBJECT: "Invitation to visit our new premises"
FIRST_RECIP_OR_ORIG: "JACK STEWART"
SUBMIT_TIME: 19950426044442Z
ATTACH::
TYPE: "TXT"
FILE: "BUA_LOGIN_DIR:invite.txt"
END
END
```

Example 2

This example demonstrates the use of the *SUBMIT* command to send a message including a binary attachment. The example uses address form 1 the first primary recipient and address form 2 for the second primary recipient.

Contents of command script

```
$SUBMIT INFOBASE: SENTFOLDER
IMPDU::
  IMPDU_HEADING::
    PRIORITY: NORMAL
    ORIGINATOR::
      X400F1::
        COUNTRY:        DE
        ADMD:           OMSADMD
        PERSONAL_NAME::
          SURNAME:    "HILL"
        END
      END
    END
    PRIMARY_RECIPIENTS::
      X400F1::
        COUNTRY:        DE
        ADMD:           OMSADMD
        PRMD:           OMSPRMD
        PERSONAL_NAME::
          SURNAME:    "STEWART"
        END
        DELIVERY_NOTIFICATION:      YES
        RECEIPT_NOTIFICATION:       YES
        NON_RECEIPT_NOTIFICATION:  YES
      END
      X400F2::
        COUNTRY:        DE
        ADMD:           OMSADMD
        UNIQUE_UA_ID: 92002
      END
    END
    SUBJECT: "New P44 design data"
  END
  IMPDU_BODY::
    ATTACH::
      TYPE: BIN
      FILE: P44_draft.cad
    END
  END
END
```

Contents of result file

```
COMMAND: SUBMIT INFOBASE:SENTFOLDER
RESULT::
IPM_ID::
LOCAL_IPM_ID: "U01035-54QD3FB0B"
END
MPDU_ID: "E7BF9FF611CE7F8B00AA0D89"
SEQ_NR: 00000000000000E9
SUBJECT: "New P44 design data"
FIRST_RECIP_OR_ORIG: "JACK STEWART"
SUBMIT_TIME: 19950426044330Z
ATTACH::
TYPE: "BIN"
FILE: "BUA_LOGIN_DIR:p44_draft.cad"
END
END
```

Example 3

This example demonstrates the use of the *SUBMIT* command to send a message including a text body part. The example uses address form 3 for both the primary recipient and the copy recipient.

Contents of command script

```
$SUBMIT INFOBASE: SENTFOLDER
IMPDU::
  IMPDU_HEADING::
    PRIORITY: NORMAL
    ORIGINATOR::
      X400F1::
        COUNTRY:        DE
        ADMD:           OMSADMD
        PERSONAL_NAME::
          SURNAME:      "HILL"
        END
      END
    END
    PRIMARY_RECIPIENTS::
      X400F3::
        COUNTRY:        DE
        ADMD:           OMSADMD
        X121_ADDRESS: "0044 181 7182936"
      END
    END
    COPY_RECIPIENTS::
      X400F3::
        X121_ADDRESS: "0044 171 3669292"
        ADMD:           OMSADMD
        COUNTRY:        DE
        DELIVERY_NOTIFICATION:      NO
        RECEIPT_NOTIFICATION:       NO
        NON_RECEIPT_NOTIFICATION:  YES
      END
    END
    SUBJECT: "P44 demonstration"
  END
  IMPDU_BODY::
    TEXT:
      The demonstration of the new P44
      self-stabilizing towbar will take
      place at our premises during the
      afternoon of July 24. We shall be
      sending out a formal invitation
      in the next few days, but perhaps
      you can note the date already.
      Hope to see you there.
      Regards
      Graham
    EOTEXT
  END
END
```

Contents of result file

```
COMMAND: SUBMIT INFOBASE:SENTFOLDER
RESULT::
IPM_ID::
LOCAL_IPM_ID: "U01035-54QD3E967"
END
MPDU_ID: "CC77AD5611CE7F8B00AA0889"
SEQ_NR: 00000000000000E4
SUBJECT: "P44 demonstration"
FIRST_RECIP_OR_ORIG: "0042 2 71829365"
SUBMIT_TIME: 19950426044244Z
END
```

UPLOAD

This command transfers an EDI Transmission Set from the subscriber's directory to the subscriber's EDI folder.

Box type: EDI box.

Syntax

```
$UPLOAD [INFOBASE:<infobase_name>]
        FILE:<filename>

        [TYPE:{TXT|BIN}]
        [DELIVERY_NOTIFICATION:{YES|NO}]
        [RECEIPT_NOTIFICATION:{YES|NO}]
        [NON_RECEIPT_NOTIFICATION:{YES|NO}]
        [SUBJECT:<subject>]
```

Parameters

INFOBASE:<infobase_name>

infobase_name can contain only the value *SENTFOLDER* (default). This parameter is optional. Refer to section 3.4 on page 16 for details of this specification.

FILE:<filename>   *filename* specifies the name and location of the Transmission Set to be sent. The file must be copied to BUA_LOGIN_DIR and the filename extension must be .ETS (EDIFACT Transmission Set).

TYPE:{TXT|BIN}   This Parameter is used to specify the type of bodypart. Available types are TXT and BIN. The default is specified by the EDI-agreement.

DELIVERY_NOTIFICATION:{YES|NO}

This parameter is used to specify whether you require a delivery notification or not. *NO* is the default value.

RECEIPT_NOTIFICATION:{YES|NO}

This parameter is used to specify whether you require a receipt notification or not. *NO* is the default value.

NON_RECEIPT_NOTIFICATION:{YES|NO}

This parameter is used to specify whether you require a Non-Receipt notification or not. *NO* is the default value.

SUBJECT:<subject>   *subject* specifies the subject of the Transmission Set.

Description

The *UPLOAD* command is used to transfer an EDI Transmission Set from the subscriber's directory to the subscriber's EDI folder, thus making it available for transmission by OpenMS. The Interchange to be transferred must already have been made available to OpenMS using an appropriate file transfer protocol.

The data of the *UPLOAD* operation appears as the send date for the Interchange in the EDI folder and the entry status is set to *SENT*.

The sequence of the *DELIVERY_NOTIFICATION*, *RECEIPT_NOTIFICATION* and *NON_RECEIPT_NOTIFICATION* specifications is not fixed.

The specified file is deleted after the *UPLOAD* command has executed successfully.

Result syntax

```
RESULT::
  UPLOAD_FILE:<filename>
  UPLOAD_STATUS:{TRANSFERRED|IGNORED_BECAUSE_EMPTY|FAILED}
END
```

Result data

UPLOAD_FILE:<filename>   *filename* contains the name of the uploaded file.

UPLOAD_STATUS:{TRANSFERRED|IGNORED_BECAUSE_EMPTY|FAILED}
                                indicates the result status of the *UPLOAD* command.

Example

The *UPLOAD* command in this example transfers the EDI Transmission Set stored in the file "ORDER_324.ETS" to the subscriber's EDI folder.

The *UPLOAD* command in this example transfers all EDI Transmission Sets whose subject contains the string "ORDER 324". The transmission sets will be stored in the file "ORDERS_324.ETS".

Contents of command script
```
$ UPLOAD INFOBASE:SENTFOLDER FILE: ORDERS_324.ETS
    SUBJECT: "ORDER 324"
    DELIVERY_NOTIFICATION:     NO
    RECEIPT_NOTIFICATION:      YES
    NON_RECEIPT_NOTIFICATION: YES
```

Contents of result file
```
COMMAND: UPLOAD INFOBASE:SENTFOLDER
    FILE: ORDERS_324.ETS
    SUBJECT: "ORDER 324"
    DELIVERY_NOTIFICATION:     NO
    RECEIPT_NOTIFICATION:      YES
    NON_RECEIPT_NOTIFICATION: YES
RESULT::
UPLOAD_FILE: "BUA_LOGIN_DIR:ORDERS_324.ETS"
UPLOAD_STATUS: TRANSFERRED
END
```

Appendix A: Information for the administrator

This appendix provides information intended for the system administrator in order to ensure correct operation of the Batch User Agent. It does not include information on general tasks relating to the OpenMS system as a whole (such as starting and stopping the system and registering users). These are described in detail in the OpenMS Operator's Reference Manual and the OpenMS Operator's Guide.

Appendix A.1: Overview

An external program (daemon) periodically scans the login directories defined for the registered BUA users in order to determine whether they contain any command scripts (with the extension ".bua"). If the daemon finds such a file, it first defines the logicals (see Table 1 on page 17) and starts the BUA process using the relevant OpenMS user name and password (see section A.2). The BUA process first checks whether all the logicals it requires have been defined, then it parses the configuration file (see section A.3) and processes the command script. Once the process has finished processing the command script, it sets the exit status (see section A.4), which may or may not be evaluated by the daemon.

Appendix A.2: BUA command line syntax

The syntax used to start the BUA process is as follows:

```
omsbua <oms_user_name> <oms_password> [/in=<command_file>
/out=<result_file>]
```

The parameters "/in=" and "/out=" are optional, if defined, only the command file explicitly defined by "/in=" parameter will be processed.

Optional Parameters

command_file	specifies the name of the file containing the command script, which the Batch User Agent is to process. No path is permitted with this specification. Two file names have special meanings:
sys$input	(VMS only) reads input from standard input (used for test purposes only)
<result_file>	specifies the name of the file to which the results are to be written. No path is permitted with this specification. Two filenames have special meanings:
sys$output	(VMS only) writes output to standard output (used for test purposes only)

Appendix A.3: BUA configuration file

During initialization, the Batch User Agent checks that the logicals it requires have been defined (see Table 1 on page 17). This also involves checking for the existence of the logical BUA_CONFIG_FILE. If this logical is defined, it must point to a valid configuration file. The configuration file is optional, and must conform to the following syntax if defined:

• Comments start with an exclamation mark (!) and extend to the end of the line.

• White-spaces outside of quoted strings are ignored and empty lines are ignored.

- String values must be enclosed in double quotes.

- Quotes in strings must be doubled.

- If the same keyword is specified more than once, all but the last values are ignored.

- The default values are used for keywords, which are not specified.

- The general syntax is as follows:
  `<keyword> = <value> <eol>`
  However, string values may extend over several lines.

In the event of an error during parsing of the configuration file, an error is issued to standard error and parsing continues.

The following keywords are currently supported:

Keyword	Purpose	Type	Range/length	Default
send_events	send events to event logger	enum	ON/OFF	ON
print_errline	display input line where error occurred	enum	ON/OFF	ON
loginbanner	login banner	string	1 to 1023	Hello!
logoutbanner	logout banner	string	1 to 1023	Good Bye!

Table 9: Keywords permitted in the configuration file

The following shows a sample configuration file:

```
! BUA_CONFIG_FILE

!

send_events   = OFF

loginbanner   = "   Welcome to OMS V22-10"

logoutbanner  =

"=======================================================

+      see you later alligator                        +
+                                                      +
+                 .....in a while crocodile            +

======================================================="
```

! END_OF_CONFIG_FILE

Appendix A.4: BUA exit status

The Batch User Agent returns an exit status in the symbol OMS_BUA_ERROR_SEVERITY. The daemon that started the process then uses this code to initiate any activities, which may be required in the event of an error.

OMS_BUA_ERROR_SEVERITY can take the following values:

  0   BUA stopped with warning

  1   BUA stopped successfully

2   BUA stopped on error
This code is returned in particular if an invalid user name is specified, if the user is not active, if a license error occurs or if an internal error occurs.

8   BUA stopped on temporary error - RETRY
This code is returned in particular if the account is currently in use.

Appendix B: System limits and PROFILE parameters

OpenMS does not predefine values for the following limits. The limits are either imposed by the server system or set by the system administrator.

Maximum value for	For submission	For retrieval
number of recipients	Recipients-limit in profile	-
number of body parts	Attachments-limit in profile	-
nesting depth for attached messages	N/A	25
size of all body parts	Size-limit in profile	-
number of stored messages	not applicable	- [×]
number of messages processed by *LIST, DELETE, FETCH* or *DOWNLOAD* (selector limit)	not applicable	Message-limit in profile
number of addresses returned by *LIST INFOBASE:SUBSCRIBER* (search limit)	not applicable	Address-limit in profile

Table 10: System limits

The user profile determines the behavior of the Batch User Agent. The settings in the profile cannot be modified from the Batch User Agent. They can, however be modified by the administrator or by the user if he or she logs into the Local User Agent and uses the MODIFY PROFILE command. The current values for the PROFILE parameters can be displayed with the READ PROFILE command in the Local User Agent.

Note:

Any changes to the PROFILE parameters made using the Local User Agent will also apply to the Batch User Agent.

[×] Response times will be slower if the number of messages stored is very large.

The following PROFILE parameters are relevant to the Batch User Agent:

Field name	Meaning	Legal values
Displayed with the SHORT qualifier:		
Alternate-recipient	Is message forwarding allowed?	YES, NO
Archive-time-hours	How long sent messages must be archived	0 = do not archive; otherwise 1 - 8760
Content-type	Default message content type for *SUBMIT*	IPM84, IPM88
Hide-recipients	*FETCH INFOBASE: <infobase_name> HEADER* shows only your name (*YES*) or also the names of all recipients (*NO*)	YES, NO (default)
Implicit-conversion	Is automatic character set conversion allowed?	YES, NO
Textbodypart	Default format for non-binary body parts	TEXT, ISO LATIN 1
Timezone	Your local time zone	As defined by the system operator
Extra fields displayed with the DETAIL qualifier:		
Message-limit	Maximum selectable number of messages	an integer
Address-limit	Maximum number of messages returned by *LIST SUBSCRIBER*	an integer
Attachments-limit	Maximum number of body parts	an integer
Recipients-limit	Maximum number of message recipients	an integer (0 = message submission not allowed; you can only receive incoming mail)
Store-sent	Enables/disables storage of submitted messages	YES, NO
Size-limit	Maximum size of message body	an integer
Boxtype	Your mailbox type	EDIBOX, MAILBOX
Purge-time-hours	Time after which messages in *MAILBOX* and *SENT-FOLDER* are automatically deleted.	an integer (0 = automatic deletion is disabled)
Service-class	Defines the class of service you are allowed to use.	ALL, LOCALONLY

Table 11: Fields displayed by the READ PROFILE command

Appendix C: Error messages

This appendix lists the Batch User Agent error messages in alphabetical order. All error messages are written to the result file of a BUA command script. Refer to section 4.3 on page 21 for a description of the result file syntax.

The structure of an BUA error is as following:

```
ERROR::
ERROR_ID: %BUATXT-S-abbreviation of error text    abbr. of ERROR_TEXT
ERROR_NUMBER: nnnnnnnnn                            number for internal use only
ERROR_TEXT: error text.                            short description of error
ERROR_INFO::                                       explanation of error, e.g.
additional error information.                       input line number,  etc.
END
END
```

Note: If there is an error section in your output file of BUA processing always look at the ERROR_ID (not the ERROR_NUMBER) to check it with the described errors below.

The list below is sorted by the ERROR_ID, the actual error message (ERROR_TEXT) is shown in bold. This is followed by an explanation. The response describes possible steps to correct the error.

%BUATXT-S-ACCTINUSE

Sorry, your account is already in use

Explanation
A command script file has been sent to the Batch User Agent during an open User Agent (e.g. LUA) session.

Response
Log out from the User Agent and send the command script to the Batch User Agent again.

%BUATXT-S-ADDR_ERR

Invalid address form error

Explanation

The specified address form is invalid.

Response
Specify another address form.

%BUATXT-S-CURR_CONTEXT_ERR

Current context is not defined

Explanation
The CURRENT qualifier has been specified in a command, but there is no current object because the session has just started and no object has yet been selected to be the current object.

Response
Specify an object using a selector.

%BUATXT-S-DEFER_DEL_ERR

Defer date specified is invalid

Explanation
An invalid defer date has been specified.

Response
Specify a correct date.

%BUATXT-S-DELIV_REP_ERR

Object is delivery report error

Explanation
An error occurred during generation of a delivery report.

Response
Please contact the operations personnel to correct this problem.

%BUATXT-S-FILE_ERR

File error

Explanation
An object with the entry status NEW has been specified in the FILE command. New messages cannot be moved from the INFOLDER infobase until they have been read.

Response
Read the message, then move it to another folder.

%BUATXT-S-FOLD_ERR

Folder error

Explanation
A user-defined folder has been specified, but this folder does not exist in the mail cabinet.

Response
Use the LIST FOLDER command to get a list of all existing folders in the mail cabinet and specify another folder name.

%BUATXT-S-FORW_ERR

Object cannot be forwarded error

Explanation
The specified address is incorrect.

Response
Check that the recipient exists and observe the rules for specifying the address.

%BUATXT-S-INT_ERR

Internal error

Explanation
An internal error, e.g. a communication problem between programs, has occurred.

Response
Please contact the operations personnel to correct this problem.

%BUATXT-S-INVUSERN

Invalid user name error

Explanation
An invalid user name was used.

Response
Use a correct user name.

%BUATXT-S-MAX_ATT_ERR

Max attachments exceeded error

Explanation
A maximum of 25 attachments can be added to a message. The specified message has more than 25 attachments.

Response
Reduce the number of attachments and try again.

%BUATXT-S-NEST_ATT_ERR

Attachment too deeply nested error

Explanation
Attachments can be nested only to a maximum depth of 25.

Response
Restrict the nesting depth of attachments to 25.

%BUATXT-S-OBJ_DEL_ERR

Object is already deleted error

Explanation
The specified object has already been deleted and the command cannot act on this object any more.

Response
None.

%BUATXT-S-OBJ_EMP_ERR

Object is empty error

Explanation
The specified object is empty. It may have been deleted during a Local User Agent session. Commands specified in the command script cannot act on this object any more.

Response
Please contact the operations personnel to correct this problem.

%BUATXT-S-OBJ_NEW_ERR

Object is new error

Explanation
The specified object has the entry status NEW and cannot therefore be filed or sent.

Response
Read the message before filing it.

%BUATXT-S-OBJ_OLD_ERR

Object is old error

Explanation
The specified message has the entry status OLD, and the command cannot therefore act on this message.

Response
Use a different entry status for selection.

%BUATXT-S-PARAM_ERR

Invalid parameters error

Explanation
The specified parameter is invalid.

Response
See the list of valid parameters in chapter 5.2 and specify the correct parameter.

%BUATXT-S-RANGE_ERR

Range error

Explanation
There are no objects in the specified range or the range has been specified for an invalid infobase or user-defined folder.

Response
Check the objects using the LIST FOLDER command and specify a new range or specify a valid infobase.

%BUATXT-S-READREPORT_ERR

Read report error

Explanation
An error occurred during generation of a read report.

Response
Please contact the operations personnel to correct this problem.

%BUATXT-S-RECIP_ERR

Recipient improperly defined

Explanation
The address format has not been specified correctly.

Response
Check the address format and try again.

%BUATXT-S-SEMANTIC_ERROR

Semantic error

Explanation
A semantic error has been detected in the command script. For example, the parameter INFOBASE has not been specified, or the entry status DRAFT has been specified with INFOLDER, or a command has been used for an EDI box, which is only valid for a MAIL box or vice versa.

Response
Check the command syntax in your script and try again.

%BUATXT-S-SEQ_NR_ERR

Sequence number error

Explanation
A sequence number that does not exist has been specified.

Response
Check the existing sequence numbers using the LIST command and select a valid sequence number.

%BUATXT-S-SERV_ERR

Service error

Explanation
A command specific to MAIL boxes has been specified for an action on the EDI box or vice versa.

Response
See the list of valid commands in chapter 5.2 and try again.

%BUATXT-S-SYNTAX_ERROR

Syntax error

Explanation
An unexpected input was found after a valid command.

Response
See the related command description, correct the input and try again.

%BUATXT-S-VAL_RECIP_ERR

No valid recipient

Explanation
The specified recipient does not exist or the wrong address has been specified.

Response
Check that the recipient exists and observe the rules for specifying the address.

EDI error report

Sample output:

```
IPMSGID:
     ID: E00003216-000000
ORIGINATOR:
     COUNTRY: DE
     ADMD: OMSADMD
     PRMD: OMSPRMD
     SURNAME: OMSCLSND1
     ORGNAME: OMSE-Edi
SUBJECT: EDI Error Report
TEXT(1):
        Subject              : EDI Error Report
                               Unable to process EDI document
        Reason               : *** invalid interchange syntax
***
        Cont Reference       : cr1
        Sender ID/Type       : EDICLSND1:Q1
        Receiver ID/Type     : EDIOPREC1:Q1
        EDI Parse Error
          TS fileposition    : 300
          IC number          : 1
          IC segment         : 4
          IC position        : 11
          IC segment code    :
```

Subject:	always remains the same.
Reason:	gives the reason, why there occurred an EDI error.
Cont Reference:	contains the number of the control reference.
Sender ID/Type:	contains the Logonname of the sender and, if it exists, separated by a colon the TP Qualifier.
Receiver ID/Type:	contains the Logonname of the receiver and, if it exists, separated by a colon the TP Qualifier.
EDI Parse Error:	contains tracking-information on the occurred error

Appendix D: Non-delivery reason and diagnostic codes

The return values for reason and diagnostic codes as defined in X.411 are as follows:

Reason codes

Return value	Argument	Description
00000000	transfer-failure	Indicates that, while the MTS was attempting to deliver or probe delivery of the subject-message, some communication failure prevented it from doing so.
00000001	unable- to- transfer	Indicates that, due to some problem with the subject itself, the MTS could not deliver or probe delivery of the subject-message.
00000002	conversion- not-performed	Indicates that a conversion necessary for the delivery of the subject-message could not (or cannot) be performed.
00000003	physical-rendition-not-performed	Indicates that the PDAU was unable to physically render the subject-message.
00000004	physical-delivery-not-performed	Indicates that the PDS was unable to physically deliver the subject-message.
00000005	restricted-delivery	Indicates that the recipient subscribes to the restricted-delivery element-of-service (as defined in Recommendation X.400) which prevented (or would prevent) the delivery of the subject-message.
00000006	directory-operation-unsuccessful	Indicates that the outcome of a required directory operation was unsuccessful.

Table 12: Non-delivery reason codes

Diagnostic codes

Return value	Argument	Description
00000000	unrecognized-OR-Name	The recipient-name argument of the subject does not contain an OR-name recognized by the MTS.
00000001	ambiguous-OR-Name	The recipient-name argument of the subject identifies more than one potential recipient (i.e., is ambiguous).
00000002	MTS-congestion	The subject could not be progressed, due to congestion in the MTS
00000003	loop-detected	The subject was detected looping within the MTS.
00000004	recipient-unavailable	The recipient MTS-user was (or would be) unavailable to take delivery of the subject-message.
00000005	maximum-time-expired	The maximum time for delivering the subject-message, or performing the subject-probe, expired.

Return value	Argument	Description
00000006	encoded-information-types-unsupported	The encoded-information-types of the subject-message are unsupported by the recipient MTS-user.
00000007	content-too-long	The content-length of the subject-message is too long for the recipient MTS-user to take delivery (exceeds the deliverable-maximum- content-length).
00000008	conversion-impractical	A conversion required for the subject-message to be delivered is impractical.
00000009	implicit-conversion-prohibited	A conversion required for the subject-message to be delivered has been prohibited by the originator of the subject.
0000000A	implicit-conversion-not-subscribed	A conversion required for the subject-message to be delivered has not been subscribed to by the recipient.
0000000B	invalid-arguments	One or more arguments in the subject were detected as being invalid.
0000000C	content-syntax-error	A syntax error was detected in the content of the subject-message (not applicable to subject-probes).
0000000D	size-constraint-violation	Indicates that the value of one or more parameter(s) of the subject violated the size constraints defined in the X.411 Recommendation, and that the MTS was not prepared to handle the specified value(s).
0000000E	protocol-violation	Indicates that one or more mandatory argument(s) were missing from the subject.
0000000F	content-type-not-supported	Indicates that processing of a content-type not supported by the MTS was required to deliver the subject-message.
00000010	too-many-recipients	Indicates that the MTS was unable to deliver the subject-message due to the number of specified recipients of the subject-message.
00000011	no-bilateral-agreement	Indicates that delivery of the subject-message required a bilateral agreement where no such agreement exists.
00000012	unsupported-critical-function	Indicates that a critical function required for the transfer or delivery of the subject- message was not supported by the originating-MTA of the report.
00000013	conversion-with-loss-prohibited	A conversion required for the subject-message to be delivered would have resulted in loss of information; conversion with loss of information was prohibited by the originator of the subject.
00000014	line-too-long	A conversion required for the subject message to be delivered would have resulted in loss of information because the original line length was too long.
00000015	page-split	A conversion required for the subject-message to be delivered would have resulted in loss of information because an original page would be split.

Return value	Argument	Description
00000016	pictorial-symbol-loss	A conversion required for the subject-message to be delivered would have resulted in loss of information because of a loss of one or more pictorial symbols.
00000017	punctuation-symbol-loss	A conversion required for the subject-message to be delivered would have resulted in loss of information because of a loss of one or more punctuation symbols.
00000018	alphabetic-character-loss	A conversion required for the subject-message to be delivered would have resulted in loss of information because of a loss of one or more alphabetic characters.
00000019	multiple-information-loss	A conversion required for the subject-message to be delivered would have resulted in multiple loss of information.
0000001A	recipient-reassignment-prohibited	Indicates that the MTS was unable to deliver the subject-message because the originator of the subject prohibited redirection to a recipient - assigned-alternate-recipient.
0000001B	redirection-loop-detected	The subject-message could not be redirected to an alternate-recipient because that recipient had previously redirected the message (redirection-loop).
0000001C	DL-expansion-prohibited	Indicates that the MTS was unable to deliver the subject-message because the originator of the subject prohibited the expansion of DLs.
0000001D	no-DL-submit-permission	The originator of the subject (or the DL of which this DL is a member, in the case of nested DLS) does not have permission to submit messages to this DL.
0000001E	DL-expansion-failure	Indicates that the MTS was unable to complete the expansion of a DL.
0000001F	physical-rendition-attributes-not-supported	The PDAU does not support the physical-rendition-attributes requested.
00000020	undeliverable-mail-physical-delivery-address-incorrect	The subject-message was undeliverable because the specified recipient postal-OR-address was incorrect.
00000021	undeliverable-mail-physical-delivery-office-incorrect-or-invalid	The subject-message was undeliverable because the physical-delivery-office identified by the specified recipient postal-OR address was incorrect or invalid (does not exit).
00000022	undeliverable-mail-physical-delivery-address-incomplete	The subject-message was undeliverable because the specified recipient postal-OR-address was incompletely specified.
00000023	undeliverable-mail-recipient-unknown	The subject-message was undeliverable because the recipient specified in the recipient postal-OR-address was not known at that address.

Return value	Argument	Description
00000024	undeliverable-mail-recipient-deceased	The subject-message was undeliverable because the recipient specified in the recipient postal-OR-address is deceased.
00000025	undeliverable-mail-organization-expired	The subject-message was undeliverable because the recipient organization specified in the recipient postal-OR-address has expired.
00000026	undeliverable-mail-recipient-refused-to-accept	The subject-message was undeliverable because the recipient specified in the recipient postal-OR-address refused to accept it.
00000027	undeliverable-mail-recipient-did-not-claim	The subject-message was undeliverable because the recipient specified in the recipient postal-OR-address did not collect the mail.
00000028	undeliverable-mail-recipient-changed-address-permanently	The subject-message was undeliverable because the recipient specified in the recipient postal-OR-address has changed address permanently (Tmoved'), and forwarding was not applicable.
00000029	undeliverable-mail-recipient-changed-address-temporarily	The subject-message was undeliverable because the recipient specified in the recipient postal-OR-address has changed address temporarily (T on travel'), and forwarding was not applicable.
0000002A	undeliverable-mail-recipient-changed-temporary-address	The subject-message was undeliverable because the recipient specified in the recipient postal-OR-address had changed temporary address (Tdeparted'), and forwarding was not applicable.
0000002B	undeliverable-mail-new-address-unknown	The subject-message was undeliverable because the recipient has moved and the recipient's new address is unknown.
0000002C	undeliverable-mail-recipient-did-not-want-forwarding	The subject-message was undeliverable because delivery would have required physical-forwarding which the recipient did not want.
0000002D	undeliverable-mail-originator-prohibited-forwarding	The physical forwarding required for the subject-message to be delivered has been prohibited by the originator of the subject-message.
0000002E	secure-messaging-error	The subject could not be progressed because it would violate the security-policy in force.
0000002F	unable-to-downgrade	The subject could not be transferred because it could not be downgraded (see Annex B to Recommendation X.419).

Table 13: Non-delivery diagnostic codes

Appendix E: Numeric and printable strings

There are restrictions on the type and length of strings, which can be entered in the various commands.

Name	Allowed characters	Length	
		min.	max.
ADMD	numeric string (output only) or	1	16
	printable string	1	16
COMMON_NAME	printable string	1	64
COUNTRY	numeric string (output only) or	1	3
	printable string	2	2
DDA _TYPE	printable string	1	8
DDA _VALUE	printable string	1	128
FREEFORMNAME	teletex string	1	64
GENERATION	printable string	1	3
GIVENNAME	printable string	1	16
INITIALS	printable string	1	5
ORGNAME	printable string	1	64
ORGUNIT	printable string	1	32
PRMD	numeric string (output only) or	1	16
	printable string	1	16
SUBJECT	teletex string	$1^{(\times)}$	128
SURNAME	printable string	1	40
TELEPHONENUMBER	printable string	1	32
UA_CONT_ID	numeric string	1	32
UNIQUE_UA_ID	numeric string	1	32
X121_ADDRESS	numeric string	1	15

Table 14: Allowed numeric and printable strings

Values are compressed (sequences of whitespaces are reduced to a single blank) and trimmed (leading and trailing whitespaces are removed) for all values other than numeric, DDA_VALUE, ORGUNIT and SUBJECT.

[×] The minimum length for the subject is 0 in the selector.

A single blank can be specified for ADMD. If you do this, you must enclose the specification in double quotes.

Appendix E.1: Numeric strings

A numeric string is a string entirely made up of characters from the Numeric String character set. The Numeric String character set consists of the following symbols:

Symbols	Description
0 1 2 3 4 5 6 7 8 9	digits
SPACEBAR	space

Table 15: Numeric String character set

Appendix E.2: Printable strings

A printable string is a string entirely made up of characters from the Printable String character set. The Printable String character set consists of the following symbols:

Symbols	Description
A - Z	uppercase letters
a - z	lowercase letters
0 1 2 3 4 5 6 7 8 9	digits
SPACEBAR	space
'	single quote
(	left parenthesis
)	right parenthesis
+	plus sign
-	hyphen
,	comma
.	period
/	slash
:	colon
=	equal sign
?	question mark

Table 16: Printable String character set

Appendix E.3: IA5 character set

The IA5 character set (International Alphabet Number 5) consists of the following characters:

	0	1	2	3	4	5	6	7
0	NUL	DLE	SP	0	@	P	`	p
1	SOH	DC1	!	1	A	Q	a	q
2	STX	DC2	"	2	B	R	b	r
3	ETX	DC3	#	3	C	S	c	s
4	EOT	DC4	¤	4	D	T	d	t
5	ENQ	NAK	%	5	E	U	e	u
6	ACK	SYN	&	6	F	V	f	v
7	BEL	ETB	'	7	G	W	g	w
8	BS	CAN	(	8	H	X	h	x
9	HT	EM	)	9	I	Y	i	y
A	LF	SUB	*	:	J	Z	j	z
B	VT	ESC	+	;	K	[	k	{
C	FF	IS4	,	<	L	\	l		
D	CR	IS3	-	=	M	]	m	}
E	SO	IS2	.	>	N	^	n	~
F	SI	IS1	/	?	O	_	o	DEL

Table 17: IA5 character set (international reference version)

Appendix E.3.1 conversion from ISO Latin 1 to IA5IRV

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	?	DLE		0	@	P	`	p	?	?	?	?	A	?	a	?
1	SOH	DC1	!	1	A	Q	a	q	?	?	?	?	A	N	a	n
2	STx	DC2	"	2	B	R	b	r	?	?	$	?	A	O	a	o
3	ETx	DC3	#	3	C	S	c	s	?	?	$	?	A	O	a	o
4	EOT	DC4	$	4	D	T	d	t	?	?	$	?	A	O	a	o
5	ENQ	NAK	%	5	E	U	e	u	?	?	$	?	A	O	a	o
6	ACK	SYN	&	6	F	V	f	v	?	?			?	?	O	?	o
7	BEL	ETB	'	7	G	W	g	w	?	?	?	?	C	x	c	?
8	BS	CAN	(	8	H	X	h	x	?	?	"	?	E	?	e	?
9	tab	EM	)	9	I	Y	i	y	?	?	?	?	E	U	e	u
a	LF	SUB	*	:	J	Z	j	z	?	?	?	?	E	U	e	u
b	pag	ESC	+	;	K	[	k	{	?	?	<	>	E	U	e	u
c	FF	?	,	<	L	\	l			?	?	?	?	I	U	i	u
d	ret	?	-	=	M	]	m	}	?	?	-	?	I	Y	i	y
e	SO	?	.	>	N	^	n	~	?	?	?	?	I	?	i	?
f	SI	?	/	?	O	_	o	DEL	?	?	-	?	I	?	i	y

Table 18: Result from conversion to IA5IRV text.

Appendix E.3.2 conversion from ISO Latin 1 to DECmcs

The conversion to DECmcs keep the most character as they were in ISO Latin 1, a few other character can not be displayed and are mapped to '?':

160	166	168	
172	173	174	175
180	184		
190			
208	215	221	222
240	253	254	

Appendix E.3.3 IBMPC character set

For IBMPC character set normally the code page 850 is used.

(see in appr. Manual for single codes)

Appendix E.4: ISO Latin 1 character set

The right half of the ISO Latin 1 character set consists of the following characters:

	8	9	A	B	C	D	E	F
0				°	À	Ð	à	ð
1			¡	±	Á	Ñ	á	ñ
2			¢	²	Â	Ò	â	ò
3			£	³	Ã	Ó	ã	ó
4			¤	´	Ä	Ô	ä	ô
5			¥	µ	Å	Õ	å	õ
6			¦	¶	Æ	Ö	æ	ö
7			§	·	Ç	×	ç	÷
8			¨	¸	È	Ø	è	ø
9			©	¹	É	Ù	é	ù
A			ª	º	Ê	Ú	ê	ú
B			«	»	Ë	Û	ë	û
C			¬	¼	Ì	Ü	ì	ü
D			-	½	Í	Ý	í	ý
E			®	¾	Î	Þ	î	þ
F			‾	¿	Ï	ß	ï	ÿ

Table 19: ISO Latin 1 character set

The left half of the ISO Latin 1 character set contains the characters described in the IA5 character set (international reference version), except those, marked in Table 18 with a '?'. For further information please refer to the appropriate X.400 Standard, describing these character sets.

Appendix F: File formats supported on VMS

The Batch User Agent supports two different file formats. These formats are described here and are referred to in this manual as the **variable** file format and the **undefined** file format.

Variable file format

The variable file format has the following attributes as shown with the DCL command DIRECTORY/FULL:

```
File organization:   Sequential
File attributes:     Allocation: 3, Extend: 0, Global buffer count: 0
                     No version limit
Record format:       Variable length, maximum 67 bytes
Record attributes:   Carriage return carriage control
RMS attributes:      None
Journaling enabled:  None
```

Undefined

The undefined file format has the following attributes as shown with the DCL command DIRECTORY/FULL:

```
File organization:   Sequential
File attributes:     Allocation: 3, Extend: 0, Global buffer count: 0
                     No version limit
Record format:       Undefined, maximum 67 bytes
Record attributes:   None
RMS attributes:      None
Journaling enabled:  None
```

Appendix G: Message types

The User Agent entry type (*ENTRY_TYPE*) defines the type of a message. The possible values are *IPM*, *EDIM*, *RPT*, *RN* and *NRN*. These types are displayed by the *LIST* command and the *FETCH* command and can be used as message selectors.

Type	Meaning
IPM	This is a standard interpersonal message.
EDIM	This is an EDI document.
RPT	This is a delivery report.
RN	This is a receipt notification.
NRN	This is a Non-Receipt notification.

Table 20: Message type values

Glossary

ADMD

see **administrative management domain**

administrative management domain

A subdivision of the Message Transfer System that is run by a PTT authority such as the Deutsche Bundespost or British Telecom. ADMDs are required to have a routing capability to all other ADMDs.

DDA

see **domain-defined attribute**

domain-defined attribute

A service made available by a management domain (administrative or private). Naming a domain-defined attribute when addressing a message enables you to specify that your message should make use of a specific service, such as fax.

EDI

see **Electronic Data Interchange**

EDIFACT

Electronic **D**ata **I**nterchange **f**or **A**dministration, **C**ommerce and **T**ransport. See **EDI**.

Electronic Data Interchange

A set of messaging standards defining formal methods for describing the component parts of trading documents and for grouping and presenting these documents in the form of messages or trade information. EDI documents can then be exchanged between agreed Trading Partners. The EDI standard supported by the United Nations and the European Commission is **EDIFACT** (ISO Standard 9735).

Interchange

A component part of an **EDI** trading document.

MTA

see **Message Transfer Agent**

Message Transfer Agent

The part of the Message Transfer System that is responsible for actually conveying your message to its intended destination.

private management domain

A subdivision of the Message Transfer System, which is run by a non-PTT organization.

PRMD

see **private management domain**

Index